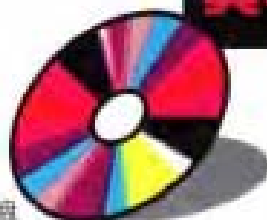


实用 化工计算机模拟

——MATLAB在化学工程中的应用



黄华江 编著



附光盘



化学工业出版社
教材出版中心

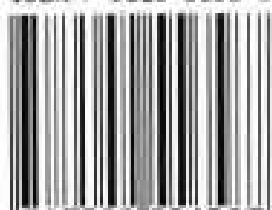
实用化工计算机模拟

——MATLAB在化工工程中的应用



附光盘

ISBN 7-5025-5658-3



9 787502 556587 >

ISBN 7-5025-5658-3/G·1477 定价: 38.00元

实用化工计算机模拟
—MATLAB 在化学工程中的应用

**Practical Computer Simulation
of Chemical Processes**

**—MATLAB's Application in
Chemical Engineering**

黄华江 编著



化学工业出版社
教材出版中心

· 北 京 ·

(京)新登字 039 号

图书在版编目(CIP)数据

实用化工计算机模拟——MATLAB 在化学工程中的应用 / 黄华江编著. —北京: 化学工业出版社, 2004.6
ISBN 7-5025-5658-3

I. 实… II. 黄… III. 化学工程—计算机辅助计算—软件包, MATLAB IV. TQ015.9

中国版本图书馆 CIP 数据核字 (2004) 第 051546 号

实用化工计算机模拟
——MATLAB 在化学工程中的应用

黄华江 编著

责任编辑: 何 丽

文字编辑: 云 雷

责任校对: 蒋 宇

封面设计: 关 飞

★

化学工业出版社
教材出版中心 出版发行

(北京市朝阳区惠新里 3 号 邮政编码 100029)

发行电话: (010) 64982530

<http://www.cip.com.cn>

★

新华书店北京发行所经销

北京永鑫印刷有限责任公司印刷

三河市海波装订厂装订

开本 787mm×1092mm 1/16 印张 19½ 字数 482 千字

2004 年 7 月第 1 版 2004 年 7 月北京第 1 次印刷

ISBN 7-5025-5658-3/G·1477

定 价: 38.00 元

版权所有 违者必究

该书如有缺页、倒页、脱页者, 本社发行部负责退换

序

黄华江博士于 1998 年获得学位, 后留在华东理工大学联合化学反应工程研究所工作。完成题为“壁冷式固定床反应器的控制系统”的博士论文过程, 他主要依靠自己的探索, 也可以说由此开始, 他一直在从事数学模型化和计算机模拟方面的研究工作, 并且其后为化工类专业的硕士生开设了“MATLAB 在化学工程中的应用”这一选修课。他的课程在教学实践中深受硕士生们的欢迎, 其原因主要是他引导硕士生们踏入了一个新的领域, 应用这一用途十分广泛、功能非常强大、具有开放式结构的数值计算软件, 可以简便地解决今天存在的大量化学工程的数值计算问题, 也可以为明天可能出现的诸多问题提供一个很好的工作平台。

本书凝聚了黄华江博士边研究、边教学、边实践的经验和体会, 我认为是一本很有价值的教材和参考书。近 20 年来, MATLAB 在西方国家得到了愈来愈广泛的应用, 并且也因此受到了相应的重视, 其原因是它使用面广, 对于几乎所有的非线性问题的数值计算都表现出实用、简捷和可靠, 因而已经在相当程度上取代了曾经一度十分流行的 C++ 等编程工具。对于化学工程中的大量数值计算问题, MATLAB 正好能发挥其特长, 很有效地予以解决。

黄华江博士的专业背景是化学工程。以他熟悉的专业背景来研究 MATLAB 的应用, 并且通过他自己研究和教学的实践来完成本书的写作, 这从书中大量的计算实例可以看出, 这也使本书与一般的讲述计算方法的书籍有所不同, 具有自己的特色。据了解, 这一类型的专门书籍如果不说没有, 至少在国内外都属十分少见。

在他完成本书的初稿以后, 他要我翻阅一下全书, 并为书作序。我应邀大致看了书的内容, 了解了编写的主要思路, 并与他进行了讨论。虽然我对书的内容并不很熟悉, 但是还是可以看到本书有其特色: 针对性和实用性。对于广大相关领域的硕士生和博士生以及有关领域的研究工作者和工程技术人员, 都是一本很有参考价值的书籍。因此欣然作序。

中国工程院院士 华东理工大学



2004 年 5 月

前 言

本书已被指定为华东理工大学研究生选修教材。出版前本书初稿已用于华东理工大学研究生选修课的教学中，课件名称为“MATLAB 在化学工程中的应用”，在首次课中，由于这门课内容很实用，选修的研究生很踊跃，其中也包括一些博士生。同学们表示对这门课很感兴趣，渴望作者尽早出书，以尽快提高他们的模拟计算能力，从而更好地帮助解决课题研究所涉及的模型化问题。作者对此深受鼓舞，经常加班加点编写。经过近一年半的努力，现在终于完成了。此书出版时，作者将原教学用书名改为“实用化工计算机模拟 — MATLAB 在化学工程中的应用”。

本书的一大特色是强调真正的实用性，这里所指的真正的实用性，就是几乎每个应用实例都包括算法及 MATLAB 函数用法、数学模型、程序说明、程序清单、计算结果等全面的求解过程。为了强调实用性，作者亲自编写了几乎所有例子的 MATLAB 源程序（共约 130 个程序，以光盘的形式配在书中），因此工作量较大。

本书在编写过程中，作者曾得到袁渭康院士和朱开宏教授的指导和鼓励，作者在此表示衷心感谢。在前期的编写准备工作期间，华东理工大学化工学院洪心进、陈茂兵和赵忠强等曾帮助录入一些文字资料，作者也在此感谢他们。作者还特别感谢父母亲在编写过程中给予的精神支持和鼓励。

尽管作者尽了最大的努力，但由于工作量大、范围广，另外作者的水平有限，不足之处在所难免，恳请读者提出宝贵意见，以使本书更加完善。

黄华江
2004 年 3 月
于上海华理园

内 容 提 要

全书共九章。第1章是化工模拟计算概述,主要叙述化工模拟的重要性、数值计算技术的发展现状、化工模拟计算文献综述等。第2章介绍 MATLAB 的编程基础,帮助读者快速 MATLAB 入门。第3章结合实例介绍常用数值计算方法及相应 MATLAB 函数的使用方法,内容包括插值与拟合、数值积分与数值微分、线性 and 非线性代数方程(组)的数值解法、常微分方程初值问题和边值问题的解法等。第4章专门介绍化工常微分方程初值问题和边值问题的应用实例,包括间歇反应器、连续槽式搅拌反应器、管式反应器、半连续反应器、传质过程、伴有反应的扩散过程、传热过程、流体流动、生化反应和过程控制等。第5章是化工中的偏微分方程及其求解,介绍有限差分法、正交配置法、MOL 法和有限元法,其中有限元法主要介绍 MATLAB 的 PDE 求解器及其求解化学工程 PDE 问题的具体方法,例子包括一维动态方程组、二维稳态方程(组)、二维动态方程等问题。第6章介绍最优化方法及其 MATLAB 常用算法,内容包括单变量最优化问题、线性规划、无约束多变量问题最优化、二次规划、多变量有约束最优化(非线性规划)问题和最小二乘法等。第7章结合实例详细介绍参数估计方法和模型辨识方法。第8章介绍化工试验设计方法及化工数据处理。第9章介绍神经网络(线性神经网络、BP 神经网络和径向基神经网络)及其相应的 MATLAB 函数,并结合实例介绍神经网络的使用方法。

本书可供化学工程、化工工艺、生化工程、环境工程、制药工程及相关专业的大学高年级本科生、硕士和博士研究生教材及参考书,也可供应用数学、过程控制等相关专业的科研人员参考。

目 录

第 1 章 化工模拟计算概述	1
1.1 化工模拟计算的重要性	1
1.2 数学模型	1
1.3 几个重要的化工软件	1
1.4 数值计算技术的发展现状和编程语言的选择	2
1.5 MATLAB 简介	3
1.6 化工模拟计算文献综述	4
1.7 本书的特点	5
1.8 本书用途及读者对象	7
参考文献	7
第 2 章 MATLAB 编程基础	9
2.1 MATLAB 的集成开发环境和基本操作	9
2.1.1 集成开发环境	9
2.1.2 基本操作	10
2.1.2.1 MATLAB 的帮助功能	10
2.1.2.2 基本操作命令	10
2.2 MATLAB 的编码基础	12
2.3 变量、常数和数据类型	12
2.3.1 变量及作用域	12
2.3.2 MATLAB 数组、向量和矩阵	12
2.3.3 数据类型	14
2.3.3.1 字符型(char)和字符串(string)	14
2.3.3.2 数值型	15
2.3.3.3 单元数组(Cell Arrays)	15
2.3.3.4 结构(Structure)	16
2.3.3.5 函数句柄	17
2.3.3.6 java 类	17
2.4 数据输出和输入	17
2.4.1 数据输出	17
2.4.2 数据输入	18
2.5 函数和过程	19
2.5.1 Script 文件	19
2.5.2 函数	20
2.6 程序流程控制	21

2.6.1	if 选择语句	21
2.6.2	switch 多重分支结构	22
2.6.3	for 循环结构	22
2.6.4	while 循环结构	23
2.6.5	try...catch 语句	23
2.6.6	continue 语句	23
2.6.7	break 和 pause 语句	23
2.6.8	return 语句	23
2.7	数值计算	23
2.8	符号运算	24
2.8.1	符号表达式的生成	25
2.8.2	符号与数值之间的转换函数	25
2.8.3	符号矩阵的创建	25
2.8.4	将数值矩阵转换为符号矩阵	26
2.8.5	符号矩阵的运算	26
2.8.6	符号微分和差分	26
2.8.7	代数方程的符号求解	26
2.8.8	常微分方程的符号求解	27
2.8.9	调用 maple 的符号计算函数	28
2.9	图形可视化	29
2.9.1	二维图形	29
2.9.2	2 ^{1/2} 维图形	31
2.9.3	三维图形	32
2.10	调试方法及程序设计实例	32
2.10.1	调试方法	32
2.10.2	程序设计实例	34
2.11	优化 MATLAB 程序代码	36
2.12	小结	38
	参考文献	40

第 3 章	MATLAB 在数值分析中的应用	41
3.1	插值与拟合	41
3.1.1	数据插值	41
3.1.2	最小二乘法拟合	43
3.1.2.1	最小二乘多项式拟合	43
3.1.2.2	用最小二乘法拟合生成样条曲线	45
3.1.3	应用实例	47
3.2	数值积分与数值微分	49
3.2.1	数值积分	49
3.2.2	数值微分	51

3.2.3 化工应用实例	53
3.2.3.1 常见的数值积分例子	53
3.2.3.2 一些常见的数值微分例子	54
3.3 代数方程(组)的数值解法	63
3.3.1 线性代数方程组的解法	63
3.3.2 非线性代数方程(组)的解法	63
3.3.3 化工应用实例	64
3.4 常微分方程(组)的数值解法	73
3.4.1 常微分方程初值问题(IVP)的数值解法	74
3.4.2 常微分方程边值问题(BVP)的数值解法	78
3.5 小结	83
习题	84
参考文献	84
 第4章 化工中的常微分方程及其求解	86
4.1 间歇反应器(Batch Reactor)	86
4.2 连续槽式搅拌反应器(CSTR)	88
4.3 管式反应器	90
4.4 半连续反应器	96
4.5 传质过程	97
4.6 伴有反应的扩散过程	105
4.7 传热过程	114
4.8 流体流动	117
4.9 生化反应	118
4.10 分离-反应一体化设备	121
4.11 过程控制	123
习题	129
参考文献	131
 第5章 化工中的偏微分方程及其求解	132
5.1 概述	132
5.1.1 偏微分方程的常见类型	132
5.1.2 偏微分方程的边界条件	133
5.1.3 偏微分方程数值方法简介	133
5.2 用有限差分法解偏微分方程	134
5.2.1 有限差分法	134
5.2.2 一维动态 PDE 模型的求解	134
5.2.3 二维稳态 PDE 方程组的求解	137
5.2.4 二维动态 PDE 模型及其他问题	144
5.3 用正交配置法解偏微分方程	144

5.4	用 MOL 法解偏微分方程	148
5.4.1	MOL 法	149
5.4.2	直接调用 MATLAB 函数 pdepe() 求解一维动态 PDE 方程 (组)	149
5.5	用有限元法解偏微分方程	158
5.5.1	MATLAB PDE 工具箱简介	158
5.5.2	PDE 工具箱可解决的化学工程问题	160
5.5.2.1	传热 (Heat Transfer)	160
5.5.2.2	扩散传递 (Diffusion Transfer) —— 传质	160
5.5.2.3	流体力学方程	161
5.5.3	利用 pdetool 求解偏微分方程	162
5.5.3.1	利用 pdetool 求解 PDE 问题的一般步骤	162
5.5.3.2	应用实例	170
5.5.4	使用 PDE 工具箱的命令行函数求解偏微分方程	174
5.6	小结	187
	习题	187
	参考文献	190
第 6 章	化工最优化方法	191
6.1	最优化方法及其常用算法	191
6.1.1	最优化方法概述	191
6.1.1.1	最优化问题的基本概念	191
6.1.1.2	最优化问题的一般形式	191
6.1.1.3	优化问题一般求解方法 ^[1]	192
6.1.1.4	MATLAB 最优化工具箱的最小化函数	192
6.1.2	单变量最优化问题	192
6.1.3	线性规划	193
6.1.4	无约束多变量问题最优化	195
6.1.5	二次规划	196
6.1.6	多变量有约束最优化 (非线性规划) 问题	198
6.1.7	多目标最优化	199
6.1.8	最小二乘法	201
6.1.8.1	最小二乘问题简介	201
6.1.8.2	线性最小二乘问题	201
6.1.8.3	非线性最小二乘问题	202
6.2	化工过程的设计优化	204
6.3	化工过程的操作优化	208
6.4	其他化工优化问题	215
6.5	全局最优化简介	216
	习题	216
	参考文献	217

第7章 参数估计和模型辨识	219
7.1 概述	219
7.2 数学模型结构	219
7.2.1 代数方程模型	219
7.2.2 微分方程模型	220
7.2.2.1 ODE 方程模型	220
7.2.2.2 PDE 方程模型	220
7.3 参数估计	221
7.3.1 一般参数估计方法	221
7.3.1.1 优化准则	221
7.3.1.2 多元线性回归	221
7.3.1.3 多响应非线性回归——多响应加权最小二乘法	222
7.3.1.4 参数估计的几个重要方面	225
7.3.2 动力学参数估计	226
7.3.2.1 化学反应动力学参数估计	226
7.3.2.2 生化反应动力学参数估计	238
7.3.3 传热参数估计	243
7.3.4 相平衡参数估计	245
7.4 模型辨识	246
习题	249
参考文献	251
第8章 化工试验设计及数据处理	253
8.1 概述	253
8.1.1 基本概念	253
8.1.2 试验设计方法	253
8.2 常用数理统计	254
8.2.1 总体和样本的统计量	254
8.2.2 方差分析中的几个数学概念	255
8.3 正交实验设计与数据处理	256
8.3.1 正交实验设计方法	256
8.3.2 正交实验结果的极差分析	256
8.3.2.1 无交互作用的正交实验	256
8.3.2.2 有交互作用的正交实验	258
8.3.3 正交实验结果的方差分析	260
8.3.3.1 无交互作用的正交实验	260
8.3.3.2 有交互作用的正交实验	262
8.4 序贯实验设计	264
8.4.1 实验设计准则	264

8.4.2 序贯实验设计步骤	265
8.5 MATLAB 实验设计函数	266
8.6 化工数据处理指南	271
参考文献	271
第 9 章 神经网络在化工中的应用	273
9.1 概述	273
9.2 神经元模型与网络结构	274
9.2.1 神经元模型	274
9.2.1.1 简单神经元	274
9.2.1.2 有一向量输入的神经元	274
9.2.1.3 传递函数	275
9.2.2 网络结构	276
9.2.2.1 单层神经网络	276
9.2.2.2 多层神经网络	276
9.3 线性神经网络	278
9.3.1 线性神经元模型	278
9.3.2 线性神经网络结构	278
9.3.3 线性神经网络的 MATLAB 求解方法	278
9.4 BP 神经网络	284
9.4.1 BP 神经元模型	284
9.4.2 BP 网络结构	284
9.4.3 BP 网络的设计	285
9.5 径向基神经网络	286
9.5.1 径向基神经网络的重要函数	286
9.5.2 径向基神经元模型	287
9.5.3 径向基函数网络的结构	287
9.5.4 径向基网络的设计	287
9.5.5 普遍化回归神经网络 (GRNN) 的设计	288
9.5.6 示例	288
9.6 神经网络工具箱 nntool GUI 求解方法	290
9.7 神经网络在化工中的应用实例	294
参考文献	296
光盘内的程序文件一览表	297

第 1 章 化工模拟计算概述

1.1 化工模拟计算的重要性

化工过程模型化是化工过程设计与分析、过程研究与开发、系统工程、模型预测、优化和控制以及生产计划和调度的基础。化工过程模拟可用于尚未建立的或已建立的过程，既可用于过程的研究、开发和设计，也可用于评估其运转情况、控制和改进^[1]。

随着化学工程、计算机软硬件及计算技术的发展，通过建立过程机理模型并进行计算机数值模拟，以便对化工过程进行设计和分析、模型预测、优化和控制等，已成为化学工程的一个重要任务。因此，模拟计算对于从事化学工程的大学生、研究生、教学与研究人员、工程师等，都是非常重要的。

1.2 数学模型

数学模型的建立是化工模拟的一个关键步骤。化工过程一般包括“三传一反”，即质量传递、热量传递、动量传递和化学反应，数学模型就是以质量平衡、热量平衡和动量平衡为基础并结合反应动力学而建立的模型方程式。

涉及化工数学模型的文献很多，其中有关化学和生化反应工程可参考国外文献[2~8]和国内文献[9~14]，传递过程可参考文献[15~21]，综合方面的数学模型可参考文献[1, 22]。此外，通过计算机检索（CA、EI、维普等）可查到相关模型的文献，这些文献有时比书籍更具参考价值。

由于篇幅所限，且本书重点在于数值求解方面，因此本书的数学模型不专设章节进行讨论，而是在各例子中加以扼要叙述。

1.3 几个重要的化工软件

时常有同学问，现在已经出现一些商用化工模拟软件（如 Aspen 软件），还有没有必要通过自己编程进行化工模拟计算？下面我将回答这个问题。

确实，市面上已出现一些化工模拟软件，其中，国际知名的化工流程模拟软件主要有麻省理工学院的 Aspen 软件和模拟科学公司的 Pro II 软件。这两个工业软件可用于工厂物料衡算和能量衡算的流程模拟。现在的一般情况是，一个大型化工厂或石油化工厂的建造，都要预先使用这类软件进行工业流程模拟。但是，由于这些软件主要针对通用性问题，软件包中不包含有对特定情况需要的专用程序模块。比如流程中某个反应器的模型，需要用户按特定情况建立此反应器模型并求解，这部分作为一个独立模块，供流程模拟专用软件调用（或嵌入流程模拟程序中）。也就是说，尽管像 Aspen 和 Pro II 这样的商用流程模拟软件通用性好、功能强大，但它们仍需要我们在其程序库中扩充、补充特定情况的模拟计算程序模块。有时，我们只需要对某个关键过程或设备进行模拟计算，这种情况商用流程模拟软件通常不能解决，只能通过自己建模、编程进行模拟计算。另外，目前的大多数国内中、小型企业，为节省资金都没有购置化工流程模拟软件，这种情况，通过自己编程进行过程模拟计算，也是可以解

决大部分工程问题的。

计算流体力学的著名软件是 Fluent, 使用该软件可以解决复杂的流体力学问题, 但是比较昂贵。因此, 对于一些相对简单的流体力学问题, 可自己编程进行模拟计算。

在科学研究领域, 由于一般涉及较新的研究课题, 多数情况需要自己编程才可解决有关新的模型问题。

综上所述, 即使有各种商用化工软件, 仍十分必要掌握通过建模编程进行化工模拟计算的方法和技能。

1.4 数值计算技术的发展现状和编程语言的选择

● 数值计算技术的发展现状

由于化工过程数学模型通常是非线性的, 一般要求进行数值求解。目前, 数理统计理论、数值计算方法、最优化方法和神经网络等实用数学的有关算法已经相当成熟, 其中一些算法 (主要是数值计算和最优化) 的 FORTRAN 代码或 C/C++ 代码也出现在一些书籍上或发布于一些网站中, 且这些公开代码往往可以免费 copy 或 download, 此外, 还出现不少的数值计算程序包或库文件, 如著名的 IMSL 数学库。各种算法的成熟稳健, 以及计算机软硬件的快速发展, 为我们进行化工模拟计算创造了有利条件。

由于常用算法已相当成熟, 通用算法函数的源代码资源很丰富而且容易获得, 因此, 进行数值计算方面的编程时, 完全没必要再去编写这些通用算法函数, 而直接把现有的通用算法函数“组装”或“集成”到自己的程序中, 这样不仅编程效率高, 而且计算比较稳定。

● 编程语言的选择

可用于数值模拟计算的编程语言有:

- BASIC/Visual Basic 6.0 (简称 VB)
- Fortran/Compac Visual Fortran 6.5 (简称 CVF)
- Pascal/Delphi
- C/C++ (Visual C++ 即 VC, 或 C++ builder)
- MATLAB

其中, VB、Delphi、VC 和 C++ builder 等是支持面向对象编程 (OOP) 的语言, CVF 也可以生成标准的用户界面。而从数值计算的角度来看, Fortran 和 C/C++ 语言以前用得较多, 其通用函数代码资源最丰富, FORTRAN 程序集和 C/C++ 语言程序集。其中, 国际上比较著名的是英国剑桥大学的 Numerical Recipes, 该程序集包括 FORTRAN 和 C/C++ 两个版本, 它包括常用数值方法和最优化方法的通用函数, 并配有说明书、例子及磁盘。文献[23]详细介绍了 Numerical Recipes C++ 版本的数值计算方法, 这是一本比较好的计算方法参考书, 也可以参考中文文献[24], 它主要根据文献[23]编写而成。在 FORTRAN 方面, 还有一个更著名的数学库就是 IMSL 库, 它包含的通用函数与英国剑桥大学的 Numerical Recipes 大同小异。早期的 IMSL 库是一个静态库, 用户程序通过与 IMSL 库的静态连接来调用 IMSL 库中的函数。现在 FORTRAN 已发展到 CVF, 它附带 IMSL 动态连接库, 可以被各种支持 OOP 的高级语言如 VC、VB 等调用, 因此可进行混合语言编程, 即用支持 OOP 的高级语言开发软件界面, 而数值计算部分调用 IMSL 动态连接库, 这样就比较方便地开发数值模拟软件。IMSL 库函数可参考 CVF 的帮助功能, 或参考文献[25]。此外, 国内也有一些程序集, 如徐士良的《C 常用算法程序集》[26]和《FORTRAN 常用算法程序集》[27]。

由于 Fortran 和 C/C++ 程序的执行效率都很高,使用这两种语言编程既可充分利用现有的代码资源,如 Numerical Recipes 和 IMSL 库,又可以提高运行速度,并且可以生成具有标准用户界面的应用软件。所以,以 CVF 或 VC 作为数值计算的编程语言是比较合适的,但是,在数值计算方面使用 MATLAB 语言编程更具有优势,因为 Fortran 和 C/C++ 语言的内部函数、Numerical Recipes 和 IMSL 库的算法函数都比较有限,且对编程能力要求较高,编程效率较低,而 MATLAB 的算法很齐全,计算功能、图形可视化功能和符号运算功能都非常强大,且简单易学,扩展性好,也支持与其他面向对象的高级语言的混合编程。它本身既是一个专用数值计算软件,又是一种编程语言,它包含 600 多个常用算法的内建函数,且代码公开,此外,还有多种非常有用的工具箱,如偏微分方程、最优化方法、数理统计、样条函数和神经网络工具箱等。所以, MATLAB 作为一种方便、快捷的科学和工程计算语言,具有 Fortran 和 C/C++ 等程序设计语言所无法比拟的优越性。

使用 MATLAB 可以完成以下几个常用软件的全部或绝大多数功能:

- Maple (符号运算, MATLAB 内嵌 Maple 5.0)
- MathCAD
- Mathematica (数学演算)
- SPSS/SAS (统计分析)
- Statistics/Origin (拟合、作图软件)

另外, MATLAB 还得到第三方公司的跟随支持(二次开发软件),即第三方公司开发基于 MATLAB 平台的软件,如 Femlab 软件。总之,从算法函数和各种工具箱、编程效率、图形可视化、程序功能扩展和程序维护性等方面, MATLAB 均具有非常明显的优势。当然, MATLAB 也有不足之处,如循环运算相对较慢,不过,这部分计算工作可以用 CVF 或 VC 实现,以加快速度,然后通过混合语言编程接口加以调用,这种充分利用各自优点而进行的混合编程,无疑是最佳的编程方案。

由于 MATLAB 的诸多优点,近年来,涉及到数值计算的国外教科书和科技图书,大多数以 MATLAB 语言为编程工具来介绍数值方法,并提供 MATLAB 例子程序。基于 MATLAB 的数值模拟计算,已经成为国内外的一种趋势。

综上所述,以 CVF 或 VC 为编程语言结合使用现有通用函数库,如 IMSL 库和 Numerical Recipes 函数库,是比较合适的数值计算编程方案。而更佳的编程方案是尽量使用 MATLAB 强大的算法函数和工具箱,当某些计算任务不能直接用 MATLAB 内部函数和工具箱函数,或者用 MATLAB 编程计算效率较低时,这部分计算任务可用 CVF 或 VC 实现,然后通过混合语言编程接口加以调用,从而提高整体运算速度。使用 MATLAB 进行数值模拟计算已经成为国内外的一种趋势。

1.5 MATLAB 简介

MATLAB 是 Mathworks 公司于 1984 年推出的数值计算软件。经过近 20 年的发展, MATLAB 已经成为该领域的佼佼者,是目前国际最流行的一种科学计算软件。其特点如下。

- 支持多平台操作系统 (Windows、Unix 等)。
- 是一种简单易学的编程语言。
- MATLAB 程序很容易维护。
- 编程效率很高。由于用户程序可直接调用大量的 MATLAB 函数,因此编程速度快。

- 用途广泛。可用于数值计算和符号计算、数据分析、工程与科学绘图、图形用户界面设计、建模和仿真、控制系统设计、数字图像信号处理以及财务工程等等。

- 功能超强。包含 600 多个常用算法内建函数，有众多面向具体应用的工具箱（如偏微分方程、最优化方法、数理统计、样条函数、神经网络工具箱等）和 simulink 仿真模块。此外，其他产品延伸了 MATLAB 的能力，包括数据采集和依靠 MATLAB 语言编程产生独立 C/C++ 代码等等。其算法函数大多由国际知名专家完成，算法稳定可靠、效率高。

- 具有开放式结构，扩展功能强。MATLAB 的开放式结构使 MATLAB 产品族很容易针对特定的需求进行扩充。

- 支持混合编程技术。提供与其他面向对象的高级语言（如 VC、VB 和 C++ 等）进行混合编程的接口。

- MATLAB 函数源代码公开，有助于用户学习和研究算法。

- 第三方公司 MATLAB 软件产品的强力支持，如 femlab，可直接求解三维 PDE 问题。

简言之，MATLAB 具有非常强大的数值计算功能、图形可视化功能和符号运算功能，且简单易学、扩展性好，可以与其他面向对象的高级语言进行混合编程。

1.6 化工模拟计算文献综述

迄今为止，国内学者编著的有关化工模拟计算的文献有[1, 28~34]。这些书大多数详细叙述数值计算方法，并结合化学工程的例子，介绍模型建立及其数值解法。对于学习算法、模型建立和编程求解都有参考价值。但是，这些书主要强调数值计算方法的算法细节，有的只强调数学模型的建立，而且多数不提供源程序，只有个别提供少量例子的 FORTRAN 源程序。而当前 MATLAB 风靡世界，国外不少专家学者编著类似的书籍时，都以 MATLAB 为计算环境（平台），并随书配套刻有 MATLAB 例子源程序的光盘，从而适应新的数值计算潮流，大大提高读者的模拟计算能力。

从国外看，早期有关化工模拟计算的书籍有[35~42]，其中文献[37, 41, 42]提供许多化工例子的 FORTRAN 源程序。近年来，国外新出版的有关化工模拟的书籍，越来越多地使用 MATLAB 作为计算环境，并提供部分例子的 MATLAB 源程序。这些书简要介绍如下。

（1）数值计算方法

— Constantinides 和 Mostoufi 的书^[43] 该书着重叙述各种数值方法的推导过程（算法细节），包括线性和非线性代数、特征值问题、有限差分、插值、微分和积分、常微分方程初值问题和边值问题、偏微分方程、线性和非线性回归分析。并介绍一些化学工程例子的实际解法，给出一部分例子的 MATLAB 源程序。该书强调数值计算方法，给出的 MATLAB 源程序都是按算法细节而编写的，而不是直接调用 MATLAB 及其工具箱中的现成函数。

（2）化工过程动态模拟和控制

— Ramirez（1997）的书^[44] 该书叙述用于现代过程控制的时域方法，介绍真实工业过程的模拟方法。

— Brosilow 和 Joseph（2002）的书^[45] 重点介绍对实际应用进行算法设计和分析，以 MATLAB 和 Simulink 作为求解工具。

— Doyle、Gatzke 和 Parker（2000）的书^[46] 重点在两个不同的单元操作——燃烧炉和二元精馏塔。内容包括：稳态分析和反馈控制，一、二阶动态系统分析、频域分析和瞬态响应分析、PID 控制、前馈控制、内模控制、模型预测控制、离散时间系统的模型化与控制。

— Bequette (1998 和 2003) 的书^[47, 48] 这两本书完整地介绍化工过程动态行为的模拟方法, 着重介绍数值方法和化工过程的动态模拟, 内容包括过程模型化、数值方法、线性和非线性系统分析等。其中, 98 版介绍了以 MATLAB 和 Simulink 为工具来求解实际问题的应用例子, 而 2003 年版在 98 年版的基础上增加 MATLAB 控制系统工具箱用于求解实际问题的应用例子。

(3) 传递过程

— Benitez (2002) 的书^[15] 主要叙述传质操作中的理论和应用, 包括对流、相际传质、吸收和汽提、精馏和液-液萃取等。

— Wilkes (1999) 的书^[16] 主要叙述化工流体力学的基本原理, 并简要介绍 MATLAB PDE 工具箱的 pdetool 用于求解流体力学问题的方法。

— Thomson (2000) 的书^[17] 叙述传递现象的基本原理, 内容包括三部分: 分子传递、对流传递和宏观计算。

(4) 化学反应工程

— Fogler (1999) 的书^[5] 重点叙述化学反应工程原理及应用分析, 主要举例反应工程的例子, 并附少量 MATLAB 源程序。

— Löwe (2001) 的书^[49] 是一本德语书, 提供一些反应工程例子的 MATLAB 源程序。

(5) 吸附平衡和吸附动力学

— Do (1998) 的书^[50] 重点叙述吸附理论, 内容包括多孔介质中的吸附平衡和吸附动力学。该作者按均相和非均相两种情况分别加以叙述, 并提供部分例子的 MATLAB 源程序。

以上这些书, 都是为化学工程学生和专业人员编写的, 每本书的侧重点都不一样。数值计算方法的书重点在于算法的细节, 对于学习算法很有帮助, 但它不介绍直接调用 MATLAB 现成算法函数的高效求解方法, 也不介绍最优化方法。化工过程动态模拟和控制方面的书, 主要涉及大量的以时间为变量的常微分方程动态模型, 一般不涉及 PDE 方程, 最优化方法也很少介绍。反应工程方面的书一般给出算法比较简单的例子源程序(如非线性代数方程、ODE 方程组), 很少叙述 PDE 方程和 nonlinear 参数估计, 最优化方法只介绍简单的单变量求导方法。此外, Do (1998) 的书^[50] 专门叙述吸附平衡和吸附动力学, 论题太专、范围较窄。

1.7 本书的特点

除第 1 章和第 2 章的 MATLAB 编程基础以外, 本书以数值方法的分类为轴线加以叙述, 其他各章依次为数值计算、常微分方程、偏微分方程、最优化、参数估计和模型辨识、化工实验设计和数据处理、神经网络等, 各章均给出大量的化工应用例子及源程序。本书之所以把第 8 章的化工实验设计和数据处理也包括在内, 是因为数学模型化需要结合实验研究, 这样才能确定模型中的未知参数。

传统的数值模拟方面的书籍, 在论述一个算法时通常包括如下内容。

- ① 数学模型;
- ② 进行有关算法的数学推导, 以得到重要的计算公式(如迭代公式);
- ③ 画出详细的程序方框图;
- ④ 结合方框图详述算法的具体计算步骤;
- ⑤ 编写该算法函数, 并调试, 得到通用的函数, 供主程序调用。

目前, 有关化学工程数值模拟方面的书籍, 尽管省略了烦琐的数学推导过程, 并增加一

些专业方面的例子，但仅省略第②步骤的大部分内容，编写思路仍然没有摆脱旧的框架。

本书最大的特色之一，就是尽量使用 MATLAB 内建函数及工具箱函数，实际省略了以上的②~⑤步，也就是说，对一个算法的描述，首先提出对应问题的数学模型，再简要介绍该模型有哪些解法，然后直接列出 MATLAB 提供的相应求解函数，并介绍该函数的用法。这种首创的编写框架，最大限度地直接使用 MATLAB 内部程序集及工具箱函数，有“拿来就用”之功效，从而大大缩短描述算法的篇幅。正是由于这个原因，使得本书有较足够的空间容纳包括 PDE、最优化方法、神经网络等在内的绝大多数对化工有用的算法，并且提供多达 130 个实例（包括程序）。这种避开算法的细节描述，完全不影响我们的模拟工作。实际上，硕士生上这门课之前，已学习了数值分析，因此不必要重新介绍算法。而对本科生而言，《化工计算机应用》课要有别于《数值分析》课，不应把大量篇幅用于算法介绍和基于 Fortran 和 C 语言之类的繁杂编程，而应着重于化工应用方面，即对实际化学工程问题进行模型建立、用 MATLAB 快速数值求解及结果分析等。如果想了解算法的内部细节，可到图书馆借阅相关书籍，数值分析、最优化方法和数理统计等这类书非常多，也容易从书店买到。当然，一些重要数值方法本书仍适当叙述，如有限差分法、正交配置法、常微分方程边值问题、连续变量的最优化方法等。

本书的另一特色是强调实用性。各章内容包括完整的数值模拟全过程，即先介绍纯数学模型及其数值解法，再详细介绍相应的 MATLAB 函数及其调用方法，并给出纯数学例子及程序代码，然后举出多个化工应用例子，且每个例子基本上都包括模型建立、程序说明、程序清单（源程序）和计算结果等。此外，各章例子的完整源程序均刻录于光盘中随书发行，以最大限度地方便读者学习。

本书的特点归纳如下。

- (1) 以 MATLAB 为计算环境，易学易用，编程效率高，程序维护方便；
- (2) 内容齐全，包括 MATLAB 编程基础、数值计算、常微分方程、偏微分方程、最优化、参数估计和模型辨识、最优实验设计、神经网络等，这些对于化学工程都很实用；
- (3) 特别强调实用性；
- (4) 例子既有广度也有深度。例子丰富，覆盖面广（参见光盘内的“程序文件一览表”），而且有足够的难度和深度，如一维动态 PDE（偏微分方程组）方程组、二维稳态 PDE 方程组（非标准边界条件）、二维动态 PDE 方程、连续变量的最优化问题、多响应系统的加权非线性参数估计等等；
- (5) 典型例子印在书上以便阅读；
- (6) 配套光盘提供所有源程序。光盘包括所有例子的 MATLAB 源程序，可作为化工应用程序集，以方便读者学习使用。其中，有些例子经过适当修改可解决自己的模拟问题。

作者经检索发现，目前国内有关 MATLAB 的书绝大多数是一般性的 MATLAB 书，少数是自动控制与 MATLAB 相结合的书，而 MATLAB 与化工紧密相结合的书尚未发现。在国外，如前所述，同时涉及到 MATLAB 与化工两方面的书有一些[5, 15~17, 42~49]，但多数是配套某方面专著而提供 MATLAB 例子，如反应工程、传递现象、动态模拟与控制专著等。因是专著，其例子难免只涉及一部分的数值计算方法（如代数方程、常微分方程组），很少涉及偏微分方程组、最优化方法、参数估计和模型辨识、试验设计、神经网络等内容。有的书着重介绍数值方法，而不是介绍如何直接利用 MATLAB 本身功能强大的算法函数。

综上所述，本书同时覆盖数值计算（插值与拟合、数值积分与数值微分、线性和非线性

代数方程组、常微分方程初值问题和边值问题、偏微分方程组)、最优化、参数估计和模型辨识、化工实验设计及数据处理、神经网络等内容,且提供大量化工例子及其 MATLAB 源程序(附程序光盘),从这个角度来说,它是国内外第一本功能比较齐全的关于实用数值计算、化工模拟、优化、参数估计和试验设计等方面的书。

作者花了一年半左右的时间编写本书,力图使读者掌握更全面的计算机模拟技术,尽量提高读者的计算机模拟能力,使读者通过快速高效地求解模型,从而可以省下更多的时间集中于课题本质方面的研究。简言之,希望本书成为科学研究与工程开发非常实用的参考书。

1.8 本书用途及读者对象

本书具有多方面的功能和用途:

- (1) 作为《化工计算机模拟》或《化工计算机应用》的教材或参考书;
- (2) 作为《反应工程分析》、《化工系统工程》和《化工过程控制》等课程的参考书;
- (3) 作为数值计算方法、最优化方法和神经网络等方面的参考书;
- (4) 作为化工应用程序集。

本书可供化学工程与工艺、生化工程以及应用化学等专业的大学高年级本科生、硕士和博士研究生、教学与科研人员等读者使用,也可供应用数学、过程控制、环境工程和医药工程等专业人士参考。此外,该书也可供从事化工生产和设计、研究的工程技术人员参考。

全书可作为化工及相近专业的硕士和博士研究生的选修教材。也可作为化工及相近专业本科生如《化工计算机应用》等课程的教材,该书除画“*”号部分以外的内容可作其教学内容。

参 考 文 献

- 1 张建侯,许锡恩编著. 化工过程分析与计算机模拟. 北京: 化学工业出版社, 1989
- 2 Smith, J. M., Chemical Engineering Kinetics, 3rd edition, McGraw-Hill, 1981 有中译本, 王建华, 许学书, 黄世英, 刘栋昌, 江礼科译. 化工动力学. (第三版). 化学工业出版社和成都科技出版社, 1988
- 3 Octave Levenspiel. Chemical Reaction Engineering. Third Edition. John Wiley & Sons, Inc., 1999
- 4 Froment, G. F., and Bischoff, K. B. Chemical Reactor Analysis and Design, 2nd edition, Wiley, 1990.
- 5 Fogler H S. Elements of Chemical Reaction Engineering. 3rd Edition. Prentice Hall PTR, Upper Saddle River, New Jersey, 1999
- 6 Ricardson J. F. and Peacock D. G., Chemical Engineering, Vol. 3, Chemical & Biochemical Reactor & Process Control, 3rd edition, 世界图书出版社, 1994
- 7 Blanch H W. Biochemical Engineering. New York: Marcel Dekker, 1996
- 8 Shuler M. L. Bioprocess Engineering Basic Concepts. 2nd Edition. New Jersey: Prentice-Hall Inc., 2002
- 9 李绍芬主编. 反应工程. 北京: 化学工业出版社, 2000
- 10 朱炳辰主编. 化学反应工程. 北京: 化学工业出版社, 2001
- 11 朱开宏, 袁渭康编著. 化学反应工程分析. 北京: 高等教育出版社, 2002
- 12 李启兴等编. 数学模拟法. 北京: 人民教育出版社, 1981
- 13 戚以政, 汪叔雄编著. 生化反应动力学与反应器. 第二版. 北京: 化学工业出版社, 1999
- 14 张元兴, 许学书编著. 生物反应器工程. 上海: 华东理工大学出版社, 2001
- 15 Jaime Benitez. Principles and Modern Applications of Mass Transfer Operations. John Wiley & Sons, Inc., 2002
- 16 James O. Wilkes, Fluid Mechanics for Chemical Engineers. Prentice Hall PTR, 1999
- 17 William J. Thomson. Introduction to Transport Phenomena. Prentice Hall, 2000
- 18 Gordon L. Amidon, Ping I. Lee and Elizabeth M. Topp ed. Transport Processes in Pharmaceutical Systems. Marcel Dekker Inc., New York, 2000
- 19 王绍亭, 陈涛著. 化工传递过程基础. 北京: 化学工业出版社, 1987
- 20 詹金荣, 陈家镛著. 传递过程原理及应用. 北京: 冶金工业出版社, 1997

- 21 戴干策, 陈敏恒. 化工流体力学. 北京: 化学工业出版社, 1988
- 22 江体乾著. 化工数学模型. 北京: 中国石化出版社, 1999
- 23 Press W H, Teukdsky S A, Vetterling W T, Flannery B P. Numerical Recipes in C. Cambridge UK: Cambridge University Press, 1992
- 24 杨华中, 汗蕙编著. 数值计算方法与 C 语言工程函数库. 北京: 科学出版社, 1996
- 25 何光瑜, 高永利编著. Visual Fortran 常用数值算法集. 北京: 科学出版社, 2002
- 26 徐士良编著. C 常用算法程序集. 北京: 清华大学出版社, 1996
- 27 徐士良编. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- 28 王树森编. 化学工程计算方法. 北京: 化学工业出版社, 1989
- 29 周爱月. 化工数学. 北京: 化学工业出版社, 1993
- 30 潘亚明, 朱鹤孙编著. 化学与化工中的数学方法. 北京: 北京理工大学出版社, 1993
- 31 张吉瑞编著. 化工数值方法. 北京: 中国石化出版社, 1995
- 32 范勋培主编. 化学中的数值计算方法与 CAD. 上海: 上海交通大学出版社, 1996
- 33 孙岳明, 陈志明, 肖国民编著. 计算机辅助化工设计. 北京: 科学出版社, 2000
- 34 陈中亮主编. 化工计算机计算. 北京: 化学工业出版社, 2000
- 35 Leon Lapidus, Digital computer for chemical engineers, McGraw-Hill, 1962
- 36 John H. Seinfeld and Leon Lapidus. Mathematical Methods in Chemical Engineering Volume 3 Process Modeling, Estimation, and Identification. New Jersey: Prentice-Hall Inc., 1974.
- 37 Raman R., Chemical Process Computation. London: Elsevier Applied Science Publishers, 1985. 中译本, 许锡恩, 张福芝, 王保国, 吴诗华译. 化工过程计算. 北京: 化学工业出版社, 1992
- 38 V. G. Jenson and G. V. Jeffeys. Mathematical Methods in Chemical Engineering. 2nd Edition, London: Academic Press Inc., 1977. 中译本, 台德荣译. 化工数学方法. 第二版. 北京: 化学工业出版社, 1982
- 39 Finlayson Bruce A. Nonlinear Analysis in Chemical Engineering. New York: McGraw-Hill, 1980
- 40 Mark E. Davis, Numerical Methods and Modeling for Chemical Engineers, John Wiley & Sons, Inc., 1984.
- 41 Ramirez W. F., Computational Methods for Process Simulation, Butterworth Publishers, 1989.
- 42 William L. Luyben, Process modeling, simulation, and control for chemical engineers, Second edition, McGraw-Hill, Inc., 1990
- 43 Alkis Constantinides & Navid Mostoufi. Numerical Methods for Chemical Engineers with MATLAB Applications. Prentice Hall, 1999
- 44 W. Fred Ramirez. Computational Methods for Process Simulation. Butterworth-Heinemann, 1997
- 45 Coleman Brosilow & Babu Joseph. Techniques of Model-Based Control. Prentice Hall, 2002
- 46 Francis J. Doyle III, Edward P. Gatzke & Robert S. Parker. Process Control Modules: A Software Laboratory for Control Design. Prentice Hall, 2000
- 47 B. Wayne Bequette. Process Dynamics: Modelling, Analysis, and Simulation, Prentice-Hall PTR, A Simon & Schuster Company, New Jersey, 1998
- 48 B. Wayne Bequette. Process Control: Modeling, Design, and Simulation. Prentice Hall, 2003
- 49 Arno Löwe. Chemische Reaktionstechnik mit MATLAB und Simulink (Chemical Reaction Techniques with MATLAB and Simulink). Wiley-VCH Verlag GmbH, 2001
- 50 Duong D. Do. Adsorption Analysis: Equilibria and Kinetics. Imperial College Press, 1998

第 2 章 MATLAB 编程基础

2.1 MATLAB 的集成开发环境和基本操作

2.1.1 集成开发环境

MATLAB 是一个高度集成的编程环境，它主要包括命令窗口（Command Window）和代码编辑器。启动 MATLAB 后，即自动弹出命令窗口，如图 2-1 所示。

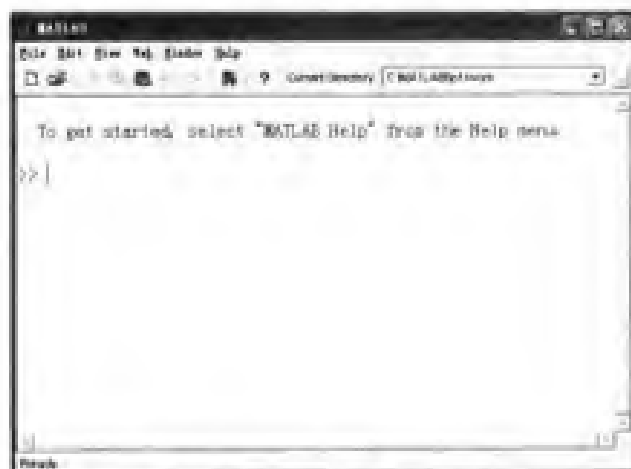


图 2-1 MATLAB 的命令窗口

图中“>>”为 MATLAB 的命令提示符。这时，可在命令窗口中直接键入 MATLAB 命令，并可观察到相应的执行结果。如键入下面命令：

```
>>a=10; b=20; c=a+b      % 回车后显示结果 c=30
```


从 File 菜单的 New 子菜单中选择执行 M-file，或单击菜单工具栏上的  按钮，则弹出代码编辑窗口（图 2-2），编程者可在此窗口输入 MATLAB 程序代码，并可进行编辑、保存、运行和调试等操作。运行程序时，其计算结果显示于命令窗口中，因此切换到命令窗口即可观察计算结果。



图 2-2 MATLAB 的编辑窗口

2.1.2 基本操作

2.1.2.1 MATLAB 的帮助功能

(1) 用 help 命令

在命令窗口中只输入 help 命令，则列出 MATLAB 搜索路径的所有目录名：

```
>>help
```

如果知道函数名，如 curvefit，则用下面命令了解其用法：

```
>>help curvefit
```

如果用目录名代替函数名，则列出目录内容，例如：

```
>>help matlab\general
```

当不知道某函数（命令）的确切名字时，先用 lookfor 命令搜索，然后用 help 命令，如

```
>> lookfor curv
```

则可从搜索到的内容中找到 curvefit 函数。

以上介绍的命令操作以及后面介绍的命令操作，请读者按介绍方法一一操练，以加深了解 MATLAB 的基本使用方法。为减少印刷版面，本章一般不列出显示结果。

(2) 联机帮助

在 MATLAB 界面中单击工具条上的问号 (?) 按钮或单击 Help 菜单中 MATLAB Help 选项，即可打开联机帮助界面。

(3) 演示帮助——看范例

```
>>demo
```

(4) PDF 帮助文档

完整的 MATLAB 软件带有 PDF 文件类型的帮助文档，每个 MATLAB 工具箱都有相应的 PDF 帮助文档，是详细的电子使用手册，可用 Adobe Acrobat Reader 软件打开进行阅读。

(5) MATLAB 网站资源

- 访问 MathWorks 公司的网站：<http://www.mathworks.com>（包括帮助文档）；
- 利用 <http://www.google.com>，以“matlab”为主题及搜索限制条件进行搜索，查到许多 MATLAB 的网站列表，然后选择访问列表中的一些网站，以获得 MATLAB 更多的相关资料。

2.1.2.2 基本操作命令

操作符 算术操作符：+ - * / ^（乘方） \（左除）

关系操作符：==（等于） ~=（不等于）

逻辑操作符：&（逻辑与） |（逻辑或） ~（逻辑否）

MATLAB 容易处理复数和无穷数：

```
>>clear, i, j % 符号 i 和 j 都默认表示虚数，当然，i 和 j 也可重新定义。Clear 是清除内存% 的命令，这里把多条命令放在一个命令行中，各条命令之间可以用逗号
```

这里以注释符“%”开头的文字用于说明左边的命令（下同）。

```
>> sqrt(-5)
```

```
>> 2/0 % 结果为 Inf，表示无穷大
```

如果表达式不能计算，MATLAB 将返回 NaN，它表示 Not-a-Number（不是一个数）：

```
>>0*log(0)
```

如果不将计算结果保存到指定变量中，则默认保存到 ans 中：

```
>>x = 3, y = 5, x*y % 计算结果为 ans = 15
```

如果不想显示结果，则命令后加分号 (;)，例如：

```
>>z = x+y; % 分号将使 z 的结果不显示出来
```

若想查看某个变量的数值，只需在命令窗口中键入变量名即可：

```
>>z
```

MATLAB 工作环境

用 who 或 whos 可观察当前工作环境中的变量：

```
>>who % 只查看变量名
```

```
>>whos % 详细了解变量名、大小、所占内存空间、数据类型等
```

用 size 命令可了解矩阵的大小：

```
>>a = [1 2 3; 4 5 6]
```

```
>>[m, n] = size(a) % m 代表行数，n 代表列数
```

用 length 命令可了解向量的大小：

```
>>v = [6 7 8]
```

```
>>p = length(v) % 这里 p 代表向量的长度（元素总数）
```

用 clear 命令可清除工作环境中的全部变量和函数，以释放相应的内存空间：

```
>>clear, who
```

也可用 clear 命令从内存中清除指定的变量：

```
>>clear x, who % 从内存清除变量 x
```

假设在工作环境（或程序）中已有一向量 x，如果经过计算后得到的新向量 x 比旧的向量 x 短，则最好在得到新向量 x 之前先用 clear 命令清除旧的向量 x。

```
>>clear all % 从内存中清除所有变量、函数以及 MEX 链接。
```

用 clc 命令可以清屏并使光标移到屏幕最左上方：

```
>>clc
```

MATLAB 所有计算都按双精度进行，但计算结果有多种显示格式，默认以 5 个数字显示，也可按格式命令（format）指定显示格式：

```
>>a=sqrt(2)
```

```
>>format short, a
```

```
>>format long e, a
```

在工作环境窗口下，用上下键（↑↓）可查看当前已键入的所有命令，当查到所要的命令时按回车键即执行一次该命令。键入某个已使用过的命令的首字母或前几个字母，再按↑键，可快速调用该命令。

在工作环境窗口（命令窗口）中可执行 DOS 和 UNIX 命令，调用格式分别是：

dos('command')和 unix('command')，如：

```
>>dos('c:'), unix('c:')
```

```
>>dos('dir'), unix('dir')
```

```
>>dos('cd C:\MATLAB6p1\toolbox'), unix('cd C:\MATLAB6p1\toolbox')
```

也可直接在命令窗口中执行一些 MATLAB 内置的类似命令：cd、dir、ls（与 dir 类似功能）、mkdir、pwd（显示当前工作目录）等。例如：

```
>>cd C:\MATLAB6p1\toolbox
```

```
>>cd 'C:\Program Files\MATLAB6p1\toolbox' % 当目录名中有空格时，要用单引号
```

读者可运行光盘中的 CommandDemo.m，以便熟悉常用的命令操作。

2.2 MATLAB 的编码基础

和其他任何编程语言一样，MATLAB 有自身的组织、编辑和格式化代码规则。

在代码中添加注释

使用注释符“%”，可以添加注释。注释可写在语句末，也可占据一整行。例如：

% 反应器尺寸

Dr = 0.025; % 反应管内径，单位：米

Hr = 1.500; % 反应管长，单位：米

将单行语句分成多行

使用续行符“...”（三个连续的小点），可以将长语句分成多行。

将多个语句放在同一行上

可以将两个或多个语句放在同一行上，但各行之间要用逗号（,）或分号（;）隔开。但是，这样做会造成程序可读性差、维护不便，所以一般单个语句单独成行。

MATLAB 的命名约定

MATLAB 代码中的变量、常数和函数的名字(标志符)，必须遵循以下规则：

(1) 以字母打头，之后是任意大小写字母、数字或下划线；

(2) MATLAB 仅使用前 31 个字符作为变量名，且大小写敏感；

(3) 不和关键字同名。关键字包括预定义语句（如 for 和 if）、内部函数名（如 exp 和 sin）、操作符（如 or 和 xor）等；

(4) MATLAB 的内部函数名必须用小写字母。例如，求矩阵 A 的逆矩 inv(A)，不能写成 INV(A)或 Inv(A)，否则会出错。

2.3 变量、常数和数据类型

2.3.1 变量及作用域

像其他编程语言那样，MATLAB 使用变量来存储数值。但是，MATLAB 变量不需要声明，可直接使用。

根据变量作用域（有效范围）的不同，可将变量分为局部变量和全局变量。在默认的情况下，函数内的变量属于局部变量，它只在函数内有效，而在该函数外部是不可用的。全局变量对于整个程序的所有过程（或函数）都有效，全局变量可用 global 关键字定义。

2.3.2 MATLAB 数组、向量和矩阵

MATLAB 数组(array)是 MATLAB 语言惟一能够处理的对象类型，MATLAB 所有类型变量，包括标量(scalar)、向量(vector)、矩阵(matrix)、字符串(string)、单元数组(cell array)、结构体(structure)和对象(object)等，都以 MATLAB 数组方式存储。

MATLAB 数组的数据存储方式：与 FORTRAN 一样，所有的 MATLAB 数据按列方式存储。例如，在命令窗口中键入如下矩阵 a：

```
>>a = ['house'; 'floor'; 'porch']
```

```
a =
```

```
house
```

```
floor
```

```

    porch
>>size(a)
    ans =
         3         5

```

但是，各个字符串的长度必须相同，否则出错，若不同，可添加空格，使之相同，如：

```
>> a = ['car      '; 'computer'; 'camera  ']
```

多维数组：维数大于 2 的 MATLAB 数组称为多维数组 (multidimensional array)。其数据存储方式与矩阵相同。

逻辑数组 (Logical Arrays)：任何非复数的数值数组或稀疏数组可以是逻辑数组，其存储方式与非逻辑数组相同。

空数组：任何类型的 MATLAB 数组可以是空数组 (empty array)，空 mxArray 至少有 1 维为零。如一个 $m = 0$ ， $n = 0$ ， $pr = \text{NULL}$ 的双精度 mxArray，是一个空数组。

向量、矩阵和多维数组：标量 (Scalar) 是由单一的值来表征的 (物理) 量；向量 (Vector) 通常是由标量的一组有序集合表示的量；数组 (array) 是元素的一维或多维的排列。数组包括一维数组 $x(i)$ 、二维数组 $x(i, j)$ 和多维数组 $x(i, j, k, \dots)$ 。矩阵 (matrix 或 matrices) 通常指矩形数组 rectangular array，即二维数组。

简言之，一个标量可以认为是一个 1×1 矩阵或者仅有一个元素的数组；一个向量是一个 $1 \times n$ (或 $n \times 1$) 矩阵，即一维数组；一个矩阵即是二维数组。

向量、矩阵的操作：由于矩阵使用最广泛，MATLAB 的基本实体是矩阵。用中括号可以引入一个矩阵，其中，一行中的元素用空格或逗号隔开，而行之间用分号 (;) 或硬回车 (换行) 分开，如：

```
>>t = [1 3 5; 2, 4, 6]
```

可对矩阵的一个元素、几个元素或子矩阵进行单独操作 (显示、调用、替换和修改等)：

```
>>t(2, 1)           % 显示一个元素
```

```
>>t(1, 3) = 7       % 元素替换
```

```
>>t(1: 2, 2: 3)     % 显示子矩阵
```

多个矩阵可结合成一个新的矩阵：

```
>>t1 = [t; t]
```

```
>>t2 = [t t]
```

转阵符号为单引号 (')：

```
>>p = t', q = [t; -4 -2 0]'
```

用冒号 (:) 产生一个行向量：

```
>>i = 1:2:10        % 两个冒号之间的数据为增量
```

由于 MATLAB 默认向量为行向量，故上面命令将显示如下结果：

```
i = 1   3   5   7   9
```

```
>>i = 1:10          % 默认增量为 1
```

```
>>p = [0:0.2:1; -1:0.4:1; 1:6]
```

冒号还常用米表示行、列或矩阵的一部分，用法如下：

```
>>p1 = p(:, 2)      % 将所有行、第 2 列的数据 (即全部第 2 列) 赋给 p1
```

```
>>p2 = p(3, :)      % 将第 3 行、所有列的数据 (即全部第 3 行) 赋给 p2
```

```
>>p3 = p(2:3, 3:5)
>>p4 = p(3, 2:end)
```

多维数组，即超过二维的数组。对矩阵 p 增加第三维：

```
>>p(:, :, 2) = ones(3,6)
```

再增加第四维：

```
>>p(1,1,1,2) = 8
```

第三维称为页，更高的维没有类似的名称。

测试向量、矩阵和数组的大小：

```
>>size(p) % 给出数组的大小
>>length(p) % 给出数组的最大长度
>>v = [2 4 6 8 10], length(v) % 给出向量 v 的长度
>>w = [1 2; 3 4; 5 6], [m,n] = size(w) % 得出矩阵 w 的行数 m 和列数 n
```

用命令 find 寻找向量或矩阵中符合指定条件的元素，返回该元素的下标：

```
>>x = [-1 -3 1 5 6], i = find(x>2), xplus = x(i) % 找出向量 x 中大于 2 的所有元素
>>X = [1 3 5; 2 4 6], [i,j] = find(X==6) % 找出矩阵 X 中元素为 6 的下标
>>Xmin = find(X==min(min(X))) % 找出矩阵 X 中最小的元素
```

一些内建的数组结构函数：

```
>>eye(2) % 产生 2×2 的单位矩阵
>>linspace(-3, 3, 5) % 在 -3~3 之间产生等间距的一个 5 元素行向量
>>logspace(-1, 3, 6) % 在 10-1~103 之间产生对数等间距的一个 6 元素行向量
>>ones(3) % 产生 3×3 的 1 矩阵
>>ones(2, 3) % 产生 2×3 的 1 矩阵
>>rand(2,3) % 产生 2×3 的随机矩阵
>>zeros(2, 3, 3) % 产生 2×3×3 的零数组
```

2.3.3 数据类型

MATLAB 的数据类型包括字符和字符串型、数值型（整型、单精度、双精度、稀疏矩阵）、单元数组、结构、java 类和函数句柄等。

2.3.3.1 字符型（char）和字符串（string）

字符型数组（character array）是常见的一种 MATLAB 数组，其元素类型是以 16 位无符号整数表示的 Unicode ASCII 码字符。其中，1×n 字符型数组，如 S='welcome'，又称为字符串（string），这与其他高级语言的字符串概念是一致的，但与 C/C++ 不同，MATLAB 字符串并不以空字符 NULL 作为结束符（结尾）。在 MATLAB 工作环境中输入 S='welcome'，再执行 whos 命令，则显示如下：

```
>> S = 'welcome'
>> whos S
```

Name	Size	Bytes	Class
S	1×7	14	char array

Grand total is 7 elements using 14 bytes

可以看出，字符串 S='welcome' 是一个 1×7 的字符数组，共占 14 个字节的存储空间。

也可用 char() 创建字符数组（字符串），如：S = char('welcome')

2.3.3.2 数值型

数值型包括整型、单精度、双精度、稀疏矩阵等。

(1) 整型

包括8位、16位和32位的有符号(int8, int16, int32)和无符号整型(uint8, uint16, uint32)。相应的类型转换函数为 int8()、int16()、int32()、uint8()、uint16()和 uint32()。

由于 MATLAB 要求所有的数学运算必须是双精度浮点运算, 因此, 整型必须先转换为双精度后再进行数学运算。例如:

```
>>a = int8([56.23 100.368])
>>b = 2*a           % 执行此命令将出错, 因为 a 为整型, 必须先转换为双精度
>>b = 2*double(a)   % 将显示 b = [112 200]
```

(2) 单/双精度浮点型及双精度复数矩阵

和其他高级语言一样, MATLAB 的浮点数有单精度和双精度之分。同整型一样, 单精度浮点必须先转换为双精度浮点后才能进行数学运算。

双精度复数矩阵 (complex matrix) 是 MATLAB 中最常用的数据类型, 其大小表示为 $m \times n$, 其中 m 和 n 分别为矩阵的行数和列数。双精度复数矩阵的实数部分和虚数部分分别用两个双精度向量来存储, 通过实部指针 pr 和虚部指针 pi 可以获取相应的数据。双精度实数矩阵的存储方式与双精度复数矩阵相同, 它实际上是双精度复数矩阵在 $pi=NULL$ (虚部指针为空) 的特例。

(3) 稀疏矩阵 (Sparse matrices)

MATLAB 矩阵有两种不同的存储方式, 即满(full)矩阵和稀疏(sparse)矩阵。通常情况下, MATLAB 矩阵为满矩阵存储类型。

稀疏矩阵是 MATLAB 矩阵的一种特殊矩阵, 它含有大量的零元素, MATLAB 只对稀疏矩阵中的那些非零元素进行存储和计算, 从而大大减少对内存的需求及相应的计算量。

稀疏矩阵除了用双精度数组 pr 和 pi 这两个参数外, 还用另外三个参数: $nzmax$, ir 和 jc 。

在 MATLAB 程序中, 稀疏矩阵并非自动生成, 而是由满矩阵转换而成, 可用函数 `sparse()` 创建, 如创建矩阵 A 的稀疏矩阵:

```
>>B = sparse(A)
```

创建稀疏矩阵后, MATLAB 遇到稀疏矩阵就会自动调用专门求解稀疏矩阵的方法, 从而提高求解效率。

稀疏矩阵的相关函数较多, 可执行“help sparfun”命令得到帮助资料。

2.3.3.3 单元数组 (Cell Arrays)

单元数组是 MATLAB 数组的一种特殊数据类型, 它用于保存不同类型和/或不同大小的数据。单元数组的每一个元素称为一个单元 (cell)。单元数组允许把不同类型的 MATLAB 数组保存在不同的单元中, 即单元数组的各个单元的数据类型可以不同。

单元数组有两种创建方式, 一种是对单元数组的各个单元直接赋值, 另一种是先用 `cell` 函数为单元数组分配空间, 然后再进行赋值。

有三种直接赋值方式

(1) 单元下标用括号“()”括起来, 而单元的内容用“{ }”括起来, 如:

```
>>clear all
>>a(1,1) = {[1 2; 3 4]};
>>a(1,2) = {[0 1]};
```



```
>>a(2,1) = {'Hello'};
```

```
>>a(2,2) = {2+3i};
```

执行以上几个命令将显示:

```
a =
```

```
    [2×2 double]    [1×2 double]
```

```
    'Hello'    [2.0000+ 3.0000i]
```

(2) 单元下标用“{}”括起来, 而赋值语句等式右边的单元内容用“[]”括起来:

```
>>a{1,1} = [1 2; 3 4];
```

```
>>a{1,2} = [0 1];
```

```
>>a{2,1} = 'Hello';    % 右边只有一个元素时可省略去“[]”
```

```
>>a{2,2} = 2+3i;
```

(3) >>a = {[1 2; 3 4], [0 1]; 'Hello', 2+3i}

显示单元数组的命令

```
>>a    % 显示单元数组 a 的信息
```

```
>>celldisp(a)    % 显示单元数组 a 的完整内容
```

先使用函数 cell() 创建空的单元数组, 然后再赋值:

```
>>b = cell(2, 3)
```

赋值方法同直接赋值方式。

对单元数组元素的操作

```
>>c = a{1, 2}    % 将单元数组 a 的{1, 2}元素赋给变量 c, 注意是“{}”, 而不是“()”。
```

2.3.3.4 结构(Structure)

结构也可以保存不同类型的数据。与 C 语言类似, MATLAB 结构用于存取相关的数据, 它由一组称为域(fields)的成员变量(向量)构成, 每一个域可以为不同的 MATLAB 数据类型。与单元数组不同, 结构把数据存于域中, 而不是存于单元中。正如标量是一个 1×1 的数使型数组一样, MATLAB 结构实际上是一个维数为 1×1 的结构数组, 它的存储方式与 1×n 单元数组(n 为域的数量)相同。

结构数组的定义有两种方法, 一种是直接赋值, 另一种使用 struct() 函数。

例: 下面建立一个名为 student 的结构, 其成员包括学生名字、专业、科目和成绩:

```
>>student.name = 'Zhang Jun';
```

```
>>student.major = 'Chemical Engineering';
```

```
>>student.subject = ['英语' ;'政治' ;'数学' ;'化工原理' ;'物理化学' ];
```

```
>>student.entrance_exam = [62 68 72 82 90];
```

```
>>student 回车后显示:
```

```
student =    name: 'Zhang Jun'
```

```
           major: 'Chemical Engineering'
```

```
           subject: '英语  政治  数学  化工原理  物理化学 '
```

```
           entrance_exam: [62  68  72  82  90]
```

这里, student 是包含一个结构的数组, 该结构有四个域。

要增加结构数组, 在结构名后面加下标, 如:

```
>>student(2).name = 'Li Xia';
```

```
>>student(2).major = 'Chemical Engineering';
>>student(2).subject = ['英语 ','政治 ','数学 ','化工原理 ','物理化学 '];
>>student(2).entrance_exam = [60 72 68 85 88];
```

使用 struct()创建结构数组的格式如下:

```
struct_array_name = structure ('field1', values1, 'field2', values2,...)
```

其中, 'field1', 'field2', ...代表域名(成员变量); values1, values2...代表对应的域值, 其值必须是大小相同的单元数组、数量单元或单个数值。

2.3.3.5 函数句柄

函数句柄是 MATLAB 的一种数据类型。创建一个函数句柄, 可用于保存函数的所有信息, 以便将来对它进行调用。函数句柄可作为参数传递给其他函数, 并与 feval 函数一起使用, 以调用该函数句柄所属的函数。

创建函数句柄的方法:

```
funhandle = @function_name
```

其中, function_name 为用户指定的函数名, funhandle 为返回的函数句柄, 可被另外的函数调用。

使用函数句柄可带来许多好处: (a) 用句柄可将一个函数传递给另一个函数; (b) 减少定义函数的文件个数; (c) 改善重复操作的性能; (d) 保证函数计算的可靠性。

有关函数句柄的应用可参见 2.10.2 节的例 2-4。执行“help FUNCTION_HANDLE”命令, 可了解函数句柄的详细信息。

2.3.3.6 java 类

java 类是为 MATLAB 提供在 MATLAB 环境下进行 java 计算的功能, 详细信息请看在线帮助。

2.4 数据输出和输入

2.4.1 数据输出

(1) 用 Save 命令

用 Save 命令可以将 MATLAB 工作环境中的几个或全部变量保存到文件中。

下面先产生数据:

```
>> clear
>> a = [1 2], b = [3 4 5], c = [-1 0; 6 8]
```

下面命令将 MATLAB 工作环境中的所有变量保存在“file1.mat”中:

```
>> save file1
```

如果只需保存其中一些变量, 可把这些变量的名字列于文件名后面:

```
>> save file2 a c
```

上面产生的文件, 扩展名默认为“.mat”, 它们只能由 MATLAB 读取, 要想在其他软件中使用数据, 可按文本方式保存:

```
>> save file3 b -ascii
```

这里, 产生一个没有扩展名的文本文件“file3”, 也可按指定扩展名(如“.txt”)保存为文本文件:

```
>> save file4.txt b -ascii
```

但按文本方式保存时，只保存数据部分，而变量名并未得到保存。因此，很少按文本方式保存，而通常按默认格式保存为.mat 文件。

(2) 用 fprintf 函数

用函数 fprintf() 可按格式将数据输出至屏幕或写格式化数据到文件中。该函数与 C 语言的同名函数功能和用法大致相同。如执行命令

```
>>x = 35; y = 68.3579; string = 'Results: ';  
>> fprintf('%t%s\tx = %5d,\ty = %8.2f', string, x, y)
```

将显示 Results: x = 35, y = 68.36

函数 fprintf() 包括两部分，即单引号括起来的部分以及单引号后面的变量表。引号内包含一些控制符用于控制后面变量表中各变量的输出格式，如上面的 fprintf 语句中，控制符包括“\t”、“%s”、“%5d”、“%8.2f”等。其中，“\t”控制符相当于按一次<Tab>键使光标向右移动一个制表位。以“%”开头后接“s”、“d”或“f”的控制符用于控制（按顺序）变量表中各变量的输出数据类型及其所占的空格数。如“%s”指示变量表中的第一个变量 string 按字符串类型输出；“%5d”指示变量表中的第二个变量 x 按整型类型输出，且共占 5 个空格；“%8.2f”指示变量表中的第三个变量 y 按浮点类型输出，且共占 8 个空格。其他的非控制符按原样显示，如“x =”、“y =”，相当于显示“x =”、“y =”。

函数 fprintf() 还可以直接显示整个向量或矩阵，这是 C 语言所没有的功能，如：

```
>>A = 0:20:120; B = 0.0:0.5:3  
>> fprintf('%t%d', A)    % 显示:    0   20   40   60   80   100   120  
>> fprintf('%t%.2f', B)    % 显示:    0.00   0.50   1.00   1.50   2.00   2.50   3.00  
MATLAB 提供的这一功能使数据输出更加简便而高效。
```

(3) 用函数 disp() 将结果输出至屏幕

例如，执行下面命令

```
>>a = [23.66   77.22   86.12];  
>>disp('The results are:')    % 显示提示信息“The results are:”  
>>disp(a)    % 只显示数据而不显示变量名:    23.6600    77.2200    86.1200
```

2.4.2 数据输入

在 MATLAB 编程中，有多种方法可以将数据输入到 MATLAB 程序。

(1) 利用 M 文件产生数据文件

利用文本编辑器可产生一个数据 M 文件，用于保存已知参数（标量、向量和矩阵）。这样，可在 MATLAB 程序中调用该数据 M 文件，将有关变量及其数据直接调入 MATLAB 内存（工作环境）中。

例如，后面例 2-3 中生成的数据 M 文件 Data_x2Dplot.m，是直接在程序编辑器中输入表 2-2 的数据并赋给矩阵变量 dat，读者可从光盘中打开此 M 文件，看看数据 M 文件是如何生成的。它是 Script 类型的 M 文件，可被其他程序直接调用。

(2) 用 Load 命令从 MAT 文件或文本文件读取数据

用 Load 命令可将指定文件的数据装载到 MATLAB 工作环境中，若装载文件扩展名为.mat，则装载时将显示以 Save 命令保存的变量。

```
>> clear  
>> load file1
```

```
>> whos
```

Name	Size	Bytes	Class
a	1x2	16	double array
b	1x3	24	double array

Grand total is 5 elements using 40 bytes

如果要装载的数据文件是文本文件，则装载数据到 MATLAB 工作环境时，全部数据将存入以该数据文件名（不包括扩展名）为变量名的向量或矩阵中。

(3) 用 fscanf 函数

用 fscanf 函数可从键盘或文件读取格式化数据，所用的控制符及用法与函数 fprintf() 完全类似。例子见光盘中的文件 xFscanf.m。

(4) 用提示输入函数 input

例如，执行到下面的命令时，程序将暂停并提示输入温度向量。用户在等式后面输入温度向量（如 210:2:230）并回车后，输入的向量即赋给变量 t：

```
>>t = input('vector of temperatures (°C) = ')
```

2.5 函数和过程

在编写程序时，一般应尽量将程序分割成较小的逻辑部件，使每个部件实现特定的功能，以简化程序设计任务。这些部件称为过程（procedure）。在 MATLAB 中，过程主要包括函数和 Script 命令，它们都可保存为 m 文件，都能在 MATLAB 工作环境中执行。其中，函数有返回值，而 Script 则没有。若用 MATLAB 编辑器，它将自动以扩展名为“.m”的文件保存。否则，保存时必须指定其扩展名为“.m”。

2.5.1 Script 文件

Script 仅仅是一连串 MATLAB 命令，它具有全局性，即一旦调用 Script 命令，Script 中的所有变量将在整个工作环境中有效。此外，当调用 Script 时，所有命令将按顺序依次被执行。当要执行一连串的命令，且这些命令需要执行多次时，创建 Script 文件命令是比较有效的。下面举例说明。

【例 2-1】 合成氨的变换反应热（ ΔH_r ）与温度 $T(K)$ 的关系如下

$$-\Delta H_r = (10000 + 0.219T - 2.845 \times 10^{-3}T^2 + 0.9703 \times 10^{-6}T^3) \times 4.184 \quad \text{kJ/kmol}$$

试编写一个 Script 文件以计算该反应热在 210~230°C 下的数值。

程序说明 在编辑器中键入下面程序清单中的命令，调试后保存为“xScript.m”文件（调试方法参见 2.9）。Script 文件开头的连续注释行，可用 help 功能查看：>>help xScript

程序清单 xScript.m

```
% An Example of Script File
clear; clc
disp('Calculating the reaction rate of ammonia synthesis')
t = input('vector of temperatures (°C) = '); T = t + 273.15; % T: Temperature, K
dH = (10000+0.219*T-2.845e-3*T.^2+0.9703e-6*T.^3)*4.184

% Plotting the results
plot(T,dH,'o-'), xlabel('T (K)'), ylabel('(-ΔH_r)'), title('Reaction rate VS temperature')
```

Script 的调用及结果

创建的 Script 命令可直接在编辑窗口或命令窗口中执行，也可在其他 Script 文件和函数中调用。要在工作环境中执行 Script 命令，只需键入：

```
>>xScript
```

这时，MATLAB 将依次执行 xScript 文件中的所有命令，执行到命令“t = input('vector of temperatures (°C) = ')"时，程序将暂停并等待用户输入温度向量。用户在等式后面输入温度向量（如 210:2:230）并回车后，即得到结果（这里从略）。

在编辑窗口运行 Script 命令的步骤是：先用 MATLAB 打开 Script 文件，即把 Script 调入编辑窗口中，再执行 Debug 菜单中 Run 命令，然后切换到命令窗口中以观察结果（若程序提示输入已知数据，先键入有关数据，然后观察到结果）。

2.5.2 函数

使用函数对解决问题更加有效，函数与 FORTRAN 的 Subroutine 类似，它通常需要输入参数、执行必要的计算，并返回计算结果。我们可以创建自定义函数，然后像 MATLAB 内部函数一样，可在工作环境中、Script 和其他函数中调用。

函数定义的一般格式：

```
function [y1, y2, ..., yn] = FuncName(x1, x2, ..., xn)    % 函数声明语句
y1 = ...          % (表达式 1)
y2 = ...          % (表达式 2)
...
yn = ...          % (表达式 n)
```

其中，输入参数为 x1, x2, ..., xn，输出参数为 y1, y2, ..., yn。各参数可以是标量、向量或矩阵。

若函数只有一个返回结果，声明语句可写为：

```
function y = FuncName(x1, x2, ..., xn)
```

若没有返回结果，声明语句写为：

```
function FuncName(x1, x2, ..., xn)    % 去掉输出参数表及等号
```

或

```
function [] = FuncName(x1, x2, ..., xn)    % 使用空的中括号“[]”
```

需要注意的是，不能在 Script 文件中定义函数，否则运行出错。

【例 2-2】 把[例 2-1]的 Script 改为函数，改好后保存为“xFunction.m”，其程序说明和清单如下。

程序说明 xFunction.m 程序第一行“function xFunction”借用 function 语句来说明程序名为 xFunction（这里 function 后面没有任何变量输出表、等号和变量输入表），程序名一般与文件名相同。按编程习惯，只用于该程序的函数通常在程序后面定义，就像 C++ 一样，如本程序中函数 ReactionHeat() 的定义置于程序后面。注意：若去掉第一行“function xFunction”，则程序变成 Script M 文件，由于该文件后面有函数定义，运行将出错（读者可试一试）。因此，内有函数声明的程序，其开头必须用 function 以不带输入输出表及等号的方式声明程序。

函数 ReactionHeat() 的第一行称为函数声明语句，该语句必须以 Function 开头，接着是输出变量表、等号、函数名和输入变量表。

M 文件开头紧连函数声明语句后面的连续注释行，可用 help 功能查看：

>>help xFunction % 查看 xFunction.m 文件开头的连续注释行

程序清单 xFunction.m

```
function xFunction                   % An Example of using Function
clear; clc
disp('Calculating the reaction rate of ammonia synthesis')
t = input('vector of temperatures (°C) = '), T = t + 273.15; dH = ReactionHeat(T)

% Plotting the results
plot(T,dH,'o-'), xlabel('T (K)'), ylabel('(-ΔH_r)'), title(' Reaction rate VS temperature')
% -----
function dH = ReactionHeat(T)
% T: Temperature, K; dH: Reaction heat (-ΔHr), kJ/kmol
dH = (10000+0.219*T-2.845e-3*T.^2+0.9703e-6*T.^3)*4.184;
```

运行程序 先在编辑窗口中打开文件 xFunction.m, 并从 Debug 菜单中选择执行 Run, 然后切换到命令窗口 (工作环境窗口) 中。在命令窗口中将看到程序提示输入温度向量的信息, 这时用户在等式后面输入温度向量 (如 210:2:230) 并回车后, 即得到结果 (这里从略)。

用函数 nargin() 和 nargout() 检索函数变量数

在自定义函数内使用函数 (或命令) nargin 和 nargout 可以确定该函数有多少个输入变量和输出变量被调用, 然后用条件语句根据变量数判断选择执行不同的任务。如:

```
function c = func(a, b)
if(nargin == 1)
    c = a.^2;       % 当调用语句只传递一个输入参数给此函数时, 执行该语句
end
if(nargin == 2)
    c = a + b;       % 当调用语句传递两个输入参数给此函数时, 执行该语句
end
```

此函数 func() 既可接受 1 个输入参数, 也可接受 2 个输入参数, 其调用示例为 xNargin.m。

函数 nargout 的用法与 nargin 相似。这两个函数的运用, 大大地提高了自定义函数的功能和灵活性, 使自定义函数更具通用性。MATLAB 大量的内部函数也正是使用这种方法。

2.6 程序流程控制

MATLAB 的程序流程控制 (flow control) 主要有 if 选择语句、switch 多重分支、for 循环、while 循环和 try...catch 等五种控制结构。

2.6.1 if 选择语句

if 语句的一般格式:

```
if condition1
    statements           % 如果 condition1 的值为 True, 则执行该语句组
elseif condition2
    statements           % 如果 condition2 的值为 True, 则执行该语句组
```

```

else
    statements          % 如果 condition1 和 condition2 都为 False，则执行该语句组
end

```

说明：①condition1 和 condition2 均为逻辑表达式；

②elseif 子句是可选的，数量也不受限制。如果缺省所有的 elseif 子句，则 if 语句变为如下格式：

```

if condition
    statements
else
    statements
end

```

③ 如果 else 子句也省略，则 if 语句变为如下格式：

```

if condition
    statements
end

```

例如：

```

x = input('x=');
if x>=0
    y = sqrt(x)
else
    y = sqrt(-x)
end

```

2.6.2 switch 多重分支结构

switch 的一般格式：

```

switch test_expr          % 测试表达式 test_expr 可以是标量或字符串
    case value
        statements        % 当 test_expr 值是 value 时，执行该语句组
    case { value1, value2,...}
        statements        % 当 test_expr 值是 value1 或 value2,或...时，执行该语句组
    otherwise,
        statements        % 当 test_expr 值都不满足前面所有情况时，执行该语句组
end

```

switch...case...otherwise 语句的能力与 if...else...end 语句类似，但对多重选择的情况，switch 语句使代码更加易读。

2.6.3 for 循环结构

for 循环用于循环次数已知情况，其一般格式如下：

```

for 循环变量 = 表达式 1(初值):表达式 2(步长):表达式 3(终值)
    statements (语句组)
end

```

for...end 循环的执行过程是：先计算初值和终值，并把初值赋给循环变量；再判断循环变量的值是否超过了终值。若超过，则退出循环，执行 end 后面的语句，否则执行循环体的

语句组，之后将循环变量加上一个步长，然后重复执行循环体内容，直至循环变量超过终值而退出循环为止。举例如下：

```
i = 0;
for x = 1:0.2:10
    i = i+1;
    y(i) = 1/x
end
```

注意 此例仅用于介绍 for 循环的用法，实际上改用以下两条语句功能相同，但效率更高：

```
x = 1:2:10
y = 1./x
```

强烈建议，尽量使用向量代替 for 循环，以大大提高运算速度。

2.6.4 while 循环结构

```
while condition (表达式)
    statements (执行语句组)
end
```

当 MATLAB 执行这个 while...end 循环时会首先测试 condition (条件表达式)。如果 condition 为 False (零)，则直接跳出循环，执行 end 后面的语句。如果 condition 的值为 True (非零)，则执行语句组 statements，然后退回到 while 语句再测试条件。

2.6.5 try...catch 语句

try...catch 是用于对异常进行处理的语句。把有可能引起异常的语句放在 try 控制块中，这样当 try 控制块中的 statement 语句引起异常时，catch 控制块就可以捕获它，并针对不同的错误类型，进行不同的处理。它与 C++ 中的 try...catch 语句作用一样。

2.6.6 continue 语句

continue 语句通常置于 for 循环或 while 循环内，根据条件执行 continue 语句。当执行 continue 时，即跳出循环体中尚未执行的语句，接着进行下一次是否进行循环的判断。

2.6.7 break 和 pause 语句

break 语句也通常置于 for 循环或 while 循环内，根据条件执行 break 语句，以直接退出最内层的 for 循环或 while 循环。而用 pause 语句将使程序暂停以等待用户按键，当用户按下任意键时程序才继续往下执行。例子参见光盘中的 xBreakPause.m。

2.6.8 return 语句

在某个函数内部执行 return 语句时，可立即退出该函数，并返回到调用它的函数，继续运行。当函数过程已完成每个任务并可直接返回时，return 是非常有用的。

return 通常放置于函数内的一个控制结构（如 if 语句）内。执行该函数时，如果符合控制结构的某个条件，则调用 return 语句终止当前运行，并返回到调用它的函数或环境。

该语句的作用与其他高级语言（如 FORTRAN）的 return 语句的作用相同。

2.7 数值计算

标量与向量、矩阵或数组相乘——将标量分别乘以向量、矩阵或数组的所有元素：

```
>>a=[1 3 5; 2 4 6], b=3*a
```

标量与向量、矩阵或数组相加减——将标量分别加到（或减去）向量、矩阵或数组的所

有元素:

```
>>a+1, a-2
```

向量、矩阵或数组相加减——当向量、矩阵或数组大小相同时才可相加或相减:

```
>>b-a, a+b
```

向量和矩阵相乘——当向量和矩阵大小匹配时才可以相乘:

```
>> v=[-1 0 1]', a*v
```

要执行元素对元素的操作运算时, 在操作符前用“.”, 分别组成点乘 (.*)、点除 (./)、点乘方 (.^) 等, 例如:

```
>>a.*b
```

```
>>b./a, 1./a
```

```
>>a.^2
```

常用的一些 MATLAB 内部函数

max()、min()、mean()、sum()、sqrt()、log()、exp()、sort()

```
>>x = [1 3 5 7 9]; X = [1 3 5 7 9; 2 4 6 8 10];
```

```
>>max(x) % 找出向量 x 中的最大元素, 这里结果为 9
```

```
>>min(x) % 找出向量 x 中的最小元素, 这里结果为 1
```

```
>>mean(x) % 计算向量 x 中所有元素的算术平均值, 这里结果为 5
```

```
>>sum(x) % 计算向量 x 中所有元素之和, 这里结果为 25
```

```
>>sum(X) % 计算矩阵 X 中各列的所有元素之和, 这里结果为[3 7 11 15 19]
```

```
>>sum(sum(X)) % 计算矩阵 X 中所有元素之和, 这里结果为 55
```

```
>>sqrt(x) % 计算向量 x 中各个元素的开方根, 这里结果为
```

```
% [1.0000 1.7321 2.2361 2.6458 3.0000]
```

```
>>sqrt(X) % 这里计算结果 (对各个元素进行开方运算) 为:
```

```
[1.0000 1.7321 2.2361 2.6458 3.0000; 1.4142 2.0000 2.4495 2.8284 3.1623]
```

log()、exp()分别为自然对数和指数的计算函数, 与 sqrt()用法一样, 均对向量或矩阵中的各个元素分别执行相应计算, 对于标量, 跟其他编程语言相同。

其他矩阵函数

```
>>C = [2 1 0; 3 2 1; 5 1 6]
```

```
>>det(C) % 求矩阵 C 的行列式
```

```
>>[V, D] = eig(C) % 求矩阵 C 的特征值矩阵 D 和特征向量矩阵 V, 其中特征值矩阵 D  
% 是以 C 的特征值为对角线元素组成的对角矩阵, 特征向量矩阵 V  
% 是一个满秩矩阵, 它的每一列是每个特征值的特征向量, 即  $C \times V =$   
%  $V \times D$ 。
```

```
>>inv(C) % 求矩阵 C 的逆阵
```

```
>>rank(C) % 求矩阵 C 的秩
```

```
>>[U, S, V] = svd(C) % 矩阵 C 的奇异值分解
```

2.8 符号运算

MATLAB 除了具有强大的数值运算和图形可视化功能以外, 还具有强大的符号运算功能。该功能集成在符号工具箱 (Symbolic Toolbox) 中, 它是 Mathworks 公司于 1993 年从加

拿大滑铁卢大学 (Waterloo University) 购买 Maple 的使用权, 并在此 Maple 的基础上开发而成的工具箱。此外, MATLAB 还保留着与 Maple 的接口, 以实现更多的符号运算功能。

2.8.1 符号表达式的生成

在 MATLAB 符号工具箱中, 符号表达式是代表数字、函数和变量的 MATLAB 字符串或字符串数组, 它不要求变量预先赋值。

字符串或字符串数组是用单引号括起来表示的, 如 $s = \text{'Hello'}$ 。因此, 符号表达式也可直接用单引号括起来表示, 如:

创建一般符号函数:

```
>>f='sin(x)+cos(x)'           % 执行后显示: f=sin(x)+cos(x)
```

创建符号代数方程:

```
>>f='a*x^2+b*x+c=0'          % 执行后显示: f=a*x^2+b*x+c=0
```

创建符号微分方程:

```
>>f='Dy-y=x'                 % 执行后显示: f=Dy-y=x
```

符号表达式也可以用 sym 命令来创建, 如:

```
>>f=sym('sin(x)+cos(x)')
```

```
>>f=sym('a*x^2+b*x+c=0')
```

另一种创建符号表达式的方法是用 syms 命令, 如:

```
>>syms x y                    % 先用 syms 命令定义 x 和 y 为自变量, 注意 x 和 y 之间用空格隔开
```

```
>>f=y*exp(x)
```

但这种方法不能用来创建符号方程。

2.8.2 符号与数值之间的转换函数

函数 digits():

```
digits(d)           % 设置有效数字个数为 d 的近似解精度
```

函数 vpa():

```
R=vpa(s)             % 返回表达式 s 在 digits 函数设置下的精度的数值解
```

```
R=vpa(s, d)          % 函数返回表达式 s 在 digits(d)精度下的数值解
```

函数 subs():

```
subs(s, old, new)    % 以 new 代替表达式 s 中 old, 其中 old 为表达式 s 中的符号变  
% 量, new 为符号或数值变量或数值表达式。
```

例如:

```
>>f='sqrt(x)'
```

```
>>f1=subs(f, x, '2')
```

```
>>vpa(f1, 5)         % 得到开方 2 的数值计算结果
```

2.8.3 符号矩阵的创建

可以用函数 sym() 创建 (类似创建数值矩阵), 如:

```
>>a=sym('[2*x^3, cos(x), exp(5*x); 6, sqrt(x), sin(x)]')
```

则显示:

```
a =
```

```
[ 2*x^3,   cos(x), exp(5*x)]
```

```
[      6,  sqrt(x),  sin(x)]
```

2.8.4 将数值矩阵转换为符号矩阵

由于数值矩阵不能直接参与符号运算，因此，必须先转换为符号矩阵。如：

```
>>a = [10.2 20.5; 50.3 68.6];
```

```
>>b = sym(a)
```

则显示：

```
b =  
[ 51/5, 41/2]  
[ 503/10, 343/5]
```

可见，转换后的符号矩阵以精确的有理形式给出。

2.8.5 符号矩阵的运算

符号矩阵的四则运算与数值矩阵的四则运算完全相同。如：

```
>>a = sym('[x+5, 6; 1/x, 2*x]')
```

```
>>b = sym('[x, 1/x; 0, 3*x]')
```

```
>>c = a/b
```

则显示：

```
c =  
[ 1/x*(x+5), 1/3/x^3*(-5+6*x^2-x)]  
[ 1/x^2, 1/3*(-1+2*x^4)/x^4]
```

2.8.6 符号微分和差分

数分和差分 diff()：

与数值微分和差分相同，符号微分和差分的函数也是 diff()，它可以自动识别是符号还是数值。如：

```
>>diff(s,'x') % 符号表达式 s 对自变量 x 的一次求导
```

```
>>diff(s,'x',n) % 符号表达式 s 对自变量 x 的 n 次求导
```

梯度函数 gradient()：

```
[FX, FY] = gradient(F) % 返回矩阵 F 的数值梯度，FX、FY 分别表示 dF/dx  
% 和 dF/dy
```

```
[FX, FY] = gradient(F, H) % 当 H 为数量时，使用 H 为各方向点间隔。
```

```
[FX, FY] = gradient(F, HX, HY) % 使用 HX 和 HY 指定点间隔。
```

多元函数的导数——雅可比矩阵函数 jacobian()：

```
jacobian(f, v) % 计算数量或向量 f 对向量 v 的 jacobian 矩阵。当 f  
% 为数量时，函数 f 返回的梯度。
```

2.8.7 代数方程的符号求解

代数方程（组）的符号求解函数为：solve()。

调用格式：

```
g = solve(eq) % eq 为引号括起来的代数方程，未知变量缺省时  
% 采用系统默认的未知变量
```

```
g = solve(eq1, eq2, ..., eqn) % eq1, eq2, ..., eqn 都是分别用引号括起来的代数  
% 方程，构成 n 个代数方程组，未知变量按默认
```

```
g = solve(eq1, eq2, ..., eqn, var1, var2, ..., varn) % 指定未知变量 var1, var2, ..., varn
```

例如:

```
>> solve('a*x^2-b*x+c=0') % 求解单变量非线性代数方程
```

则显示:

```
ans =  
[1/2/a*(b+(b^2-4*a*c)^(1/2))]  
[1/2/a*(b-(b^2-4*a*c)^(1/2))]
```

```
>>[x, y] = solve('x^2-x*y+y = 2', '3*x^2-x+5 = 0') % 求解非线性代数方程组
```

则显示:

```
x =  
[1/6+1/6*i*59^(1/2)]  
[1/6-1/6*i*59^(1/2)]  
y =  
[32/21+5/21*i*59^(1/2)]  
[32/21-5/21*i*59^(1/2)]
```

注意: 对于单个方程或方程组, 如果不能得到符号解, 则返回数值解。

对于线性代数方程, 可用右除“\”进行符号求解, 如:

```
>> solve('10*x-y-9=0', '-x+10*y-2*z-7=0', '-2*y+10*z-6=0', 'x,y,z')
```

如:

```
>> a = sym('[10,-1,0;-1,10,-2;0,-2,10]') % 注意: 符号矩阵中元素之间最好用“,”隔开
```

```
>>b = sym('[9; 7; 6]')
```

```
>> a\b
```

则显示:

```
ans =  
[473/475]  
[ 91/95]  
[376/475]
```

执行“>>sym(a)\sym(b)”与执行“>>a\b”等效。

2.8.8 常微分方程的符号求解

ODE 方程(组)的符号求解函数是 dsolve()。

调用格式: $r = \text{dsolve}(\text{'eqn1, eqn2, ...'}, \text{'cond1, cond2, ...'}, \text{'v'})$

$r = \text{dsolve}(\text{'eqn1'}, \text{'eqn2'}, \dots, \text{'cond1'}, \text{'cond2'}, \dots, \text{'v'})$

输入变量: 'eqn1, eqn2, ...' 该字符串定义 ODEs
 $\text{'cond1, cond2, ...'}$ 该字符串定义 ODEs 的初始条件
 'v' 该字符串定义自变量

在 MATLAB 中, 约定 D 表示一次微分, D2 表示二次微分, D3 表示三次微分, ..., 符号 Dy 相当于 Dy/Dt 。函数 dsolve 把 D 后面的变量当作自变量, 并默认这些变量对自变量 t 求导, 也可以指定其他的自变量。

初始条件的定义:

'y(a) = b'

'Dy(a) = b'

其中, y 是其中的一个因变量, a 和 b 为常数。

例如, 求解二阶常微分方程:
$$\begin{cases} \left(\frac{dy}{dt}\right)^2 + y^2 = 1 \\ y(0) = 0 \end{cases}$$

执行下列命令:

```
>>y = dsolve('(Dy)^2 + y^2 = 1', 'y(0) = 0')
```

则得到结果为:

```
ans =  
[sin(t)]  
[-sin(t)]
```

例如, 求解一阶常微分方程组的初值问题

$$\begin{cases} \frac{df}{dt} = f + g \\ \frac{dg}{dt} = -f + g \\ f(0) = 1, g(0) = 2 \end{cases}$$

执行下列命令可求得结果:

```
>>s = dsolve('Df = f + g', 'Dg = -f + g', 'f(0) = 1', 'g(0) = 2')
```

```
s =  
f: [1x1 sym]  
g: [1x1 sym]
```

可通过如下方式查看结果:

```
>>f = s.f           % 显示结果: f = exp(t)*(cos(t)+2*sin(t))  
>>g = s.g           % 显示结果: g = -exp(t)*(sin(t)-2*cos(t))
```

2.8.9 调用 maple 的符号计算函数

加拿大滑铁卢大学的 maple 软件擅长于符号计算, MATLAB 已将 maple 5.0 版集成在一起, 在 MATLAB 中可以直接调用 maple 命令进行符号计算。有两种调用格式:

(1) `>> maple(statement)`

其中, 参数 `statement` 是字符串, 代表 maple 的命令, 如:

```
>>maple(dsolve('(Dy)^2 + y^2 = 1', 'y(0) = 0'))
```

显示结果为:

```
ans =  
[sin(t)]  
[-sin(t)]
```

(2) `>>maple('function', arg1, arg2, ...)`

该命令调用 maple 函数库中的函数, 输入参数 `function` 是要调用的函数的名称, `arg1, arg2, ...` 是该函数的输入参数。

例如, 求指数函数 $\exp(x)$ 在 $x = 0$ 处的 6 阶泰勒展开式:

```
>> maple('readlib(mtaylor)')
```

```
>>maple('mtaylor(exp(x), [x=0], 6)')
```

显示结果为:

```
ans = 1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5
```

2.9 图形可视化

MATLAB 具有很强的图形处理功能,可以绘制二维图形、三维图形和四维表现图等,还能对图形进行着色、消隐、光照、渲染等多项处理。

2.9.1 二维图形

利用 plot 和 line 函数可以在二维平面上绘制二维图形。其中, line 是直接对图形句柄进行操作的底层绘图命令, plot 是在底层命令基础上建立起来的高层绘图命令。

Plot 函数 绘制二维线形图

下面命令绘制向量 y 对向量 x 的图形,其中 s 代表不同线型、点标和颜色的字符串,如表 2-1 所示(用“help plot”命令也可查看表 2-1 中的内容)。常见命令格式如下:

```
>> plot(x, y, s) % 绘制一条曲线
>> plot(x1, y1, s1, x2, y2, s2, ...) % 同时绘制多条曲线
```

表 2-1 线型、点标和颜色的字符串

颜 色	点 标	线 型
b 蓝 (blue)	.	小圆点
c 青 (cyan)	o	小圆圈
g 绿 (green)	x	x 形
k 黑 (black)	+	+ 形
m 洋红 (magenta)	*	* 形
r 红 (red)	s	方形
y 黄 (yellow)	d	菱形
	v	下向三角形
	^	上向三角形
	<	左向三角形
	>	右向三角形
	p	五边形
	h	六边形
	-	实线
	:	点虚线
	-.	点划线
	--	长虚线

```
>> x = [0:2:20]; y = 2*x.^2+5*x+6;
>>plot(y) % 绘制 y 对其索引的图形
>>plot(x, y) % 绘制 y 对 x 的实线图形
>>grid % 对当前坐标添加网格线
>>xlabel('x') % 在 x 轴下方添加文本
>>ylabel('y') % 在 y 轴左侧添加文本
>>title('y = 2*x^2+5*x+6') % 在图形顶部添加图的标题
```

下面命令只画出散点,并以“+”作为点标:

```
>>plot(x, y, '+')
```

用 gtext 命令可直接把文本加到图中,且文本在图中的位置由鼠标点击确定:

```
>>gtext('抛物线')
```

也可用 text 命令将文本放置于指定位置上:

```
>>text(10, 200, 'PlaceText')
```

在同一个图形中画多条曲线:

```
>> plot(x, y, 'k+-', x, 2*x.^2-3*x-20, 'bs:')
```

另一种绘制曲线的函数是 fplot:

```
>>fplot('2*sin(x)+cos(x)', [0, 2*pi]) % 引号内为函数表达式
```

要画的函数也可以是用户自定义函数，如小程序 *xFplot.m*：

```
Function xFplot                                % An example of using fplot()
fplot(@func, [0, 2*pi])

function f = func(x)
f = 2*sin(x)+cos(x)
```

轴的范围可用 `axis` 显示和修改：

```
>>axis                                % 显示当前坐标轴的刻度范围
>>axis('square')                      % 在方型框内作图
>>axis('normal')                      % 恢复到正常比例
```

也可用命令“`axis([xMin xMax yMin yMax])`”设置作图上下边界：

```
>>axis([0 20 -100 990])              % 将当前坐标轴的刻度范围改为：x=0~20, y=-100~990
```

用 `subplot` 可在不同框架 (frames) 中显示多个图形：

```
>>subplot(2,1,1), plot(x, y, 'k+-')
>>subplot(2,1,2), plot(x, 2*x.^2- 3*x-20, 'bs:')
```

这里 `subplot(i,j,k)` 表示有 *i*“行”图和 *j*“列”图，而当前图是第 *k* 个图（从右往左，从顶往底数起）。

要返回到单张图状态，则打入下面命令即可：

```
>>subplot(1,1,1)
```

用 `shg` 可把当前图形窗口显示出来：

```
>>shg
```

在继续画新的图形时，可用 `clf` 命令清除已画的图形窗口：

```
>>clf
```

用 `figure(n)` 命令可画出两个以上的图形窗口 (*n* 为正整数)：

```
>>figure                                % 创建新的图形窗口，并返回图形句柄
>>figure(3)                            % 创建新的图形窗口，其图形句柄为 3
```

其他常用的二维画图函数：

```
>>area(x, y)                          % 画面积图
>>bar([1:3])                          % 画条形图
>>loglog(x, y)                        % 双对数画图函数，x 轴和 y 轴都是对数坐标
>>semilogx(x, y)                      % 半对数画图函数，x 轴是对数坐标，而 y 轴不是对数坐标
>>semilogy(x, y)                      % 半对数画图函数，y 轴是对数坐标，而 x 轴不是对数坐标
>>polar(x, y)                          % 极坐标画图函数
```

【例 2-3】 在一反应系统中，组分 A1 和 A2 的浓度随时间的变化如表 2-2 所示，试画出浓度的动态变化曲线。

表 2-2 组分 A1 和 A2 的浓度

t/min	$C_{A1}/(\text{kmol}/\text{m}^3)$	$C_{A2}/(\text{kmol}/\text{m}^3)$	t/min	$C_{A1}/(\text{kmol}/\text{m}^3)$	$C_{A2}/(\text{kmol}/\text{m}^3)$
0	0.4000	0.2000	1.5000	0.4527	0.2185
0.5000	0.4387	0.2045	2.0000	0.4342	0.2198
1.0000	0.4559	0.2125	2.5000	0.4097	0.2161

续表

t/min	$C_{A1}/(\text{kmol}/\text{m}^3)$	$C_{A2}/(\text{kmol}/\text{m}^3)$	t/min	$C_{A1}/(\text{kmol}/\text{m}^3)$	$C_{A2}/(\text{kmol}/\text{m}^3)$
3.0000	0.3892	0.2094	6.0000	0.4264	0.2066
3.5000	0.3798	0.2023	6.5000	0.4218	0.2088
4.0000	0.3830	0.1976	7.0000	0.4116	0.2086
4.5000	0.3955	0.1964	7.5000	0.4004	0.2063
5.0000	0.4109	0.1986	8.0000	0.3924	0.2030
5.5000	0.4226	0.2027	8.5000	0.3901	0.2000

程序说明 表中数据存于 script 文件 Data_x2Dplot.m 中。程序中 ylabel 和 legend 语句用到上下标的表示方法,即在字符前面放置下划线“_”将使它变成下标,放“^”将使它变成上标。若已知数据较少,则需要先插值再画图,才能保证曲线足够光滑。有时需要先对实验数据进行回归或拟合,然后再画图。

程序清单 x2Dplot.m

```
% This example demonstrates how to plot a 2-D graphic
Data_x2Dplot % Retrieve data from the script file Data_x2Dplot.m
t = dat(:,1); CA1 = dat(:,2); CA2 = dat(:,3);
plot(t,CA1,'ko-',t,CA2,'b^-')
axis([0 t(end)+0.5 min(min(dat(:,2:3)))-0.1 max(max(dat(:,2:3)))+0.05])
xlabel('Time (min)'), ylabel('C_A_1, C_A_2 (kmol/m^3)')
legend('C_A_1','C_A_2'), title('Concentration profiles')
```

结果 图形结果如图 2-3 所示。

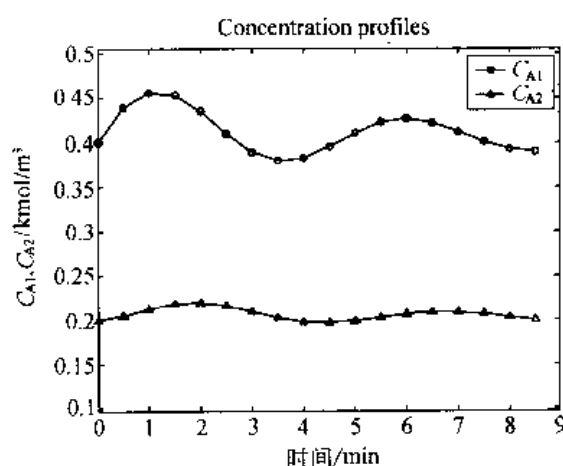


图 2-3 浓度随时间的变化

2.9.2 2^{1/2} 维图形

所谓 2^{1/2} 维图形,就是将三维图在二维坐标系统中表达。例如,已知三个向量 x 、 y 和 z , 则通过画等值线图 (contour) 使 z -水平在 x - y 坐标系统中显示出来:

% 先创建 z 矩阵:

```
>>[x, y] = meshgrid( 2:0.2:2, -2:0.2:3);
```

```
>>z = x.*exp(-x.^2 -y.^2);
```


% 绘制 z 的等值线图:

```
>>[c, h] = contour(x, y, z) % 绘制 z 的等值线图, 其中 x 和 y 指定 x 轴和 y 轴的范围
```

```
>>clabel(c, h) % 作等值线图的等值标签
```

```
>>colormap cool
```

另一种方法是从顶部 z 轴观察图形, 并把不同的颜色分配给不同的 z 值:

```
>>pcolor(x,y,z)
```

```
>>colorbar
```

```
>>shading interp
```

可用 quiver 函数绘制以箭头表示的速度向量图, 例如, 用以下命令绘制函数的梯度场:

```
>>[X, Y] = meshgrid(-2:0.2:2, -2:0.2:2); % 生成网格坐标(X, Y)
```

```
>>Z = X.*exp(-X.^2-Y.^2); % 函数值 Z = f(X, Y)
```

```
>>[DX, DY] = gradient(Z, 0.2, 0.2); % 计算梯度: DX = dZ/dX, DY = dZ/dY
```

```
>>contour(X, Y, Z)
```

```
>>hold on
```

```
>>quiver(X, Y, DX, DY) % 绘制(DX, DY)在(X, Y)处的向量图
```

```
>>grid off
```

```
>>hold off
```

2.9.3 三维图形

三维图形主要包括三维曲线和曲面图(MATLAB 又称网格面), 前者可用 plot3 函数绘制, 后者用 mesh 或 surf 绘制。

三维曲线图的绘图函数 plot3(), 是 plot 函数的三维扩展, 故其用法与 plot 函数类似:

```
>>t = 0:pi/30:5*pi; x=sin(t); y=cos(t);
```

```
>>plot3(x, y, t)
```

```
>> xlabel('x'); ylabel('y'); zlabel('t')
```

曲面图 (surfaces) 有多种绘制方法, 其中最常用的两种如下:

```
>>[x, y] = meshgrid(-pi:pi/20:pi, -pi:pi/20:pi);
```

```
>>z = cos(x).*sin(y);
```

```
>>mesh(x, y, z)
```

```
>>surf(x, y, z)
```

```
>> view(50, 50)
```

如果需要, 可以通过加阴影以美化图形:

```
>>shading interp
```

用 colorbar 可看 color scale (色度):

```
>>colorbar
```

2.10 调试方法及程序设计实例

2.10.1 调试方法

MATLAB 提供的调试器能够帮助分析程序运行是如何从过程的一部分流动到另一部分的, 分析变量是如何随着语句的执行而改变的。有了调试工具, 就能深入到程序内部去观察, 从而确定到底发生了什么以及为什么会发生。

为了更有效地使用调式手段，把可能遇到的错误分成二类：

- ① 语法错误 不符合 MATLAB 的语法规则、错误的关键字等；
- ② 运行时错误 当一个语句力图执行一个不能执行的操作时，就会发生运行的错误；
- ③ 逻辑错误 当程序未按预期方式执行时就会产生逻辑错误，即产生不正确的结果。

程序运行的正确与否，只有通过测试程序和分析产生的结果才能检验出来。

程序调试的任务就是，找出和改正语法错误和运行错误使程序正常运行，若运行结果不对，则确定导致错误结果的原因，以及错误发生的地方。

MATLAB 的调试功能包括：断点、通过代码一次经过一个语句或一个过程、进入函数或过程进行调试、显示变量的值。其调试器包括 Debug 菜单和 Breakpoints 菜单及其子菜单功能。各菜单功能说明如下。

Debug 菜单

Step	逐语句执行程序代码的下一个可执行行，但不跟踪到过程中
Step In	逐语句执行程序代码的下一个可执行行，并跟踪到过程中
Step Out (跳出)	执行当前过程的其他部分，并在调用过程的下一行处中断执行
Run	运行程序
Continue	继续运行直到程序结束
Go Until Cursor	运行到光标处
Exit Debug Mode	退出调试模式

Breakpoints 菜单

Set/Clear Breakpoints	设置/删除断点
Clear All Breakpoints	清除所有断点
Stop If Error	如果有错误就停止
Stop If Warning	如果有警告就停止
Stop If NaN Or Inf	如果计算出现 NaN (不是一个数) 或 Inf (无穷大) 就停止
Stop If All Error	

设置/删除断点的方法

在代码编辑窗口中，先用鼠标将光标定位到某一行可执行语句上，然后从 Breakpoints 菜单中选择 Set/Clear Breakpoints (或按 F12 键)，即可在此行设置/删除断点。或者，用鼠标单击要设置/删除断点的语句的左边横杠“-”上(横杠表示该行为可执行语句)。设置了断点后，横杠“-”变为红色的实心圆(断点指示器)；删除断点后，红色的实心圆又变为横杠“-”。

在断点处检查程序的方法

程序一旦运行到断点处并被中止执行，就可检查程序的当前状态(观察已执行语句中变量的数值)。具体方法是，用鼠标将光标移到变量名上，即显示该变量的当前值。或者，将变量名(或表达式)拷贝到工作环境中执行，也可以观察到当前该变量(或表达式)的当前值。如果要观察设有断点的行在运行时发生了什么，就必须至少再运行一个语句。为此要使用跟踪或单步运行。

运行程序的选定部分

如果能够识别产生错误的语句，那么单个断点就有助于对问题的定位。但更常见的情况是只知道产生错误的代码的大体区域。这时，先通过设置断点将问题区域进行隔离，然后用跟踪和单步执行来观察每个语句的效果。

单步执行

可用 Debug 菜单中的 Step 或 Step In 功能来一次一条语句地执行代码。这也称为单步执行。在单步通过每条语句之后, 可以通过查看有关变量的数值变化来判断是否存在错误。

► 单步执行代码的步骤

(1) 先在需要停止运行的地方设置一个断点。若想从程序开头逐行调试, 可在程序第一条可执行语句设置断点;

(2) 选择 Debug 菜单中的 Step 或 Step In 功能。每执行一次, 即往下执行一条语句。

绕过部分代码

当程序处在中断模式时, 可用 Go Until Cursor (运行到光标处) 功能在代码的后部选择想要停止运行的语句。因此可以掠过不感兴趣的那部分代码, 比如巨大的循环。

► 运行到光标处的步骤

(1) 把程序设置为中断模式;

(2) 把光标设置在需要停止运行的地方;

(3) 选择 Debug 菜单的 Go Until Cursor 功能。

2.10.2 程序设计实例

【例 2-4】 对于一阶方程的初值问题
$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases} \quad (1)$$

可用改进的欧拉公式求解:
$$y_1 = y_n + hf(x_n, y_n) \quad (2)$$

$$y_2 = y_n + hf(x_{n+1}, y_1) \quad (3)$$

试编程求解下面简单的初值问题:
$$\begin{aligned} y' &= -y + x + 1 \\ y(0) &= 1 \end{aligned} \quad (4)$$

程序说明 若不考虑程序的通用性, 则可直接根据迭代公式 (2) 和 (3) 编写出求解方程 (4) 的程序 IntegByEulerpro.m。但是, 如果现在需要再求解另一个或几个类似的方程, 那么要继续使用该程序必须经过相应修改, 这就比较麻烦, 且容易造成错误。为此, 下面改写成通用程序, 即根据迭代公式 (2) 和 (3) 求解方程 (1) 的通用函数 eulerpro.m。在该通用函数的定义声明中, 第一个参数是 fun, 代表方程 (1) 中的 $f(x, y)$ 函数名, 其代码部分使用函数 feval 计算 $f(x, y)$ 函数值, 如 feval(fun, x(i), y(i))。通用函数 eulerpro 编制好后, 容易编写程序 (xEulerpro.m) 调用它以求解具体的方程 (4), 注意程序 xEulerpro.m 中调用 eulerpro() 时, 给定第一个参数是 @fun (函数句柄以 @ 接函数名)。

下面简要介绍函数 feval() 的调用方法, 然后再给出程序清单。

函数 feval 的调用格式:

feval(f, x1, ..., xn) % 单响应系统, f 为函数句柄或函数名, 变量为 x1, ..., xn

[y1, ..., yn] = feval(f, x1, ..., xn) % 多响应系统

例如, 用 feval() 计算函数 $f(x, y)$ 的格式为 feval(fun, x, y), 其中, $f(x, y)$ 的定义声明如下:

function f = fun(x, y) % 这里函数名取为 fun, 当然, 也可以取为别的名字。

程序清单

IntegByEulerpro.m (直接求解方程 (4) 的程序)

```
clear all; clc
```

```
x0 = 0; xf = 1; h = 0.1;                      % x0、xf: 积分下限和上限, h: 积分步长
```

```

y0 = 1; % 初始值
n = fix((xf-x0)/h); % 积分区间[x0, xf]内的 n 等分, fix 是圆整函数
x(1) = x0; y(1) = y0;
for i = 1:n-1
    x(i+1) = x0 + i*h; y1 = y(i) + h*(-y(i)+x(i)+1);
    y2 = y(i) + h*(-y1+x(i+1)+1); y(i+1) = (y1 + y2)/2;
end
plot(x, y)

```

eulerpro.m (改进欧拉方法的通用积分函数, 用于求解通用微分方程 (1))

```

function [x, y] = eulerpro(fun, x0, xf, y0, h)
% 功能: 用改进的欧拉方法计算--阶常微分方程的初值问题
% fun: 微分方程的 f(x, y)函数句柄或函数名, x0: 积分下限, xf: 积分上限, h: 积分步长
n = fix((xf-x0)/h); x(1) = x0; y(1) = y0;
for i = 1:(n-1)
    x(i+1) = x0 + i*h; y1 = y(i) + h*feval(fun, x(i), y(i));
    y2 = y(i) + h*feval(fun, x(i+1), y1); y(i+1) = (y1 + y2)/2;
end

```

xEulerpro.m (调用函数 eulerpro 以求解方程 (4) 的例程)

```

function xEulerpro
clear all; clc
[x, y] = eulerpro(@fun, 0, 1, 1, 0.1)
plot(x, y)
function f = fun(x, y)
f = -y+x+1;

```

注: ① MATLAB 已提供许多功能强大的常微分方程 (ODE) 积分函数 (见第 3 章), 这些函数具有自动变步长的自适应功能。本例是定步长 (等步长) 的改进欧拉积分算法。

② MATLAB 及其工具箱提供大量函数, 通用性很强, 其第一个参数一般用于传递函数句柄, 在 MATLAB 中称为“function function”。因此, 调用此类函数时, 通常第一个参数给定 @function_name, 其中 function_name 是用户根据具体问题自定义的函数。如 MATLAB 函数 fzero(), roots(), ode45() 等, 详见第 3 章。

程序的基本框架构成 以上面的程序 xEulerpro.m 为例, 一个完整的程序包括:

(1) 程序的定义——在程序首行用 function 定义, 如 xEulerpro.m 中的第一行“function xEulerpro”定义程序名为 xEulerpro (按习惯, 接在 function 后面的程序名 xEulerpro 与该文件名一致), 且 xEulerpro 不带任何输入参数 (若带参数, 则变成函数定义, 而非程序定义了)。

(2) 主程序——定义程序名后, 接着就编写所谓的“主程序”代码了, 如 xEulerpro.m 的主程序代码为:

```
clear all; clc
```

```
[x, y] = eulerpro(@fun, 0, 1, 1, 0.1)
```

```
plot(x, y)
```

主程序一般包括数据输入（已知参数）、计算过程和结果输出（数值和图形结果），且计算过程往往需要调用外部函数或/和自定义的内部函数。本例无数据输入部分。

（3）程序的内部函数——在程序中定义的函数，只供该程序调用，因此称为该程序的内部函数。一般把定义这些内部函数的代码置于主程序之后。若将一个函数定义保存为一个独立文件，如上例把函数 `eulerpro()` 的定义保存为 `eulerpro.m`，以便同时被其他多个程序调用，则该函数称为外部函数。

注意 若在程序 `xEulerpro.m` 中删除第一行“function xEulerpro”，则它将出错。因为删除第一行“function xEulerpro”等于没有定义程序，即原来的程序变成脚本（Script）文件，而在脚本文件内是不允许对内部函数进行调用的。

2.11 优化 MATLAB 程序代码

为尽量加快 MATLAB 程序的运行速度，必须使 MATLAB 程序代码效率最优，具体如下：

（1）尽量避免使用循环，而使用向量或矩阵代替

MATLAB 的循环操作运算比较慢，这是它的一个不足之处。因此，应尽量避免使用循环。编程时，尽量用向量或矩阵编程，使循环向量化，从而大大提高程序执行效率。

【例 2-5】 比较循环方法和向量化方法的运行速度。

程序清单 *Speedup1.m*

```
clear all; clc
% 循环方法
tic                % 开始计时
x = 0.01;
for k = 1:1001, y(k) = log10(x); x = x + .01; end
t_cycle = toc      % 结束计时，并把所耗的时间赋给变量 t_cycle
% 向量化方法
tic, x = .01:.01:10; y = log10(x); t_vector = toc
```

程序说明 用命令 `tic` 和 `toc` 来测量某个程序段的运行时间。其中，`tic` 和 `toc` 分别用于启动和停止秒表。

运行结果 `t_cycle = 0.0500, t_vector = 0。`

（2）在循环语句之前预先对向量、矩阵或数组初始化（预分配）

在循环语句内，动态地分配向量、矩阵或数组的维数，是很浪费时间的。为此，应该在执行循环语句之前，预先用函数 `zeros()` 或 `ones()` 对向量、矩阵或数组变量进行内存分配，即初始化。对大型矩阵，更加需要预先定义维数，这样可明显地加快执行速度。

【例 2-6】 在循环语句之前预先对向量、矩阵或数组初始化的例子

程序清单 *Speedup2.m*

```
clear all; clc
% 向量、矩阵或数组的动态分配
```

```
tic
for n = 1:32, r(n) = rank(magic(n)); end
t_dynamic = toc
% 矩阵或数组的预先初始化
tic, r = zeros(32,1);
for n = 1:32, r(n) = rank(magic(n)); end
t_preinit = toc
```

(3) 尽量使用 MATLAB 的内在函数或工具箱函数

MATLAB 的内部函数或工具箱函数，是由 C 语言直接编写得到的，且很多算法由世界上最优秀的专家提供，代码由专家编写，执行速度通常比较快。因此，除非您的问题有些特殊性而找不到相应的 MATLAB 函数，否则，应尽量使用 MATLAB 提供的算法函数，以提高程序运行效率。

【例 2-7】 用下面两种方法求矩阵所有元素之和：

- ① 用自定义函数计算；
 - ② 直接使用 MATLAB 内部函数计算。
- 并比较两者的运行速度。

程序清单 *Speedup3.m*

```
function Speedup3
clear all; clc
A = rand(10, 5000);           % 随机生成一个矩阵
% 用自定义函数计算
tic, s1 = MySum(A), t1 = toc
% 直接使用 MATLAB 函数计算
tic, s2 = sum(sum(A)), t2 = toc
% -----
function s = MySum(A)          % 自定义函数功能：计算矩阵所有元素之和
s = 0; [m, n] = size(A);
for i=1:m
    for j=1:n
        s = s + A(i, j);
    end
end
end
```

运行结果 t1 = 0.2700 s, t2 = 0.0100 s。可见，直接使用 MATLAB 内部函数的计算速度要比自定义函数的运行速度快得多。

(4) 混合编程

如果已经采取了相应措施，程序执行速度仍然很慢，则可考虑与其他高级语言，如 VC、CVF 等，进行混合编程，即把 MATLAB 程序中执行速度较慢的代码部分（如循环语句），改用 VC 或 CVF 等对循环计算较快的语言进行编写，并编译成独立的可执行文件（EXE 文件、

DLL/MEX 文件或 Active X 部件), 该可执行文件能够与 MATLAB 程序之间进行数据通讯, 从而构成完整的程序。VC 或 CVF 程序与 MATLAB 程序之间最简单的数据通讯方式是通过磁盘文件进行数据交换。例如:

```
function f = loop(x, y)
save data x y
!c_loop          % 在 C 的可执行文件需要在文件名前加上感叹号“!”
load data
```

以上函数是先将变量 x 和 y 存入文件 `data.mat` 中, 在调用 C 语言编写的可执行循环程序 `c_loop` 来处理这些变量, 最后将已经处理过的变量返回到工作空间中。

详细的混合编程技术请参阅 `help` 功能或其他文献。

2.12 小结

帮助功能

`help` `demo` (范例) `lookfor` (关键词搜索) MATLAB 网站资源

操作符

算术操作符: $+$ $-$ $*$ $/$ $^$ (乘方) \backslash (左除) $.*$ $./$ $.^$ \backslash

关系操作符: $==$ (等于) \neq (不等于) \geq \leq $>$ $<$

逻辑操作符: $\&$ (逻辑与) $|$ (逻辑或) \sim (逻辑否)

注释符: $\%$ 续行符: \dots 语句之间分隔符: $,$ $;$ 转阵: $'$

冒号 ($:$) 运用: 产生一个行向量, 或表示行、列或矩阵的一部分

工作环境及内存管理

`who / whos` (查看内存变量) `size` (测量矩阵大小) `length` (测量向量大小)

`clear all` (清除内存所有变量) `clc` (清屏) `format` (设置显示格式)

单元数组 (Cell Arrays)

单元下标用“ $\{\}$ ”括起来, 如 `a{1,1} = [1 2; 3 4]`

`celldisp`: 显示单元数组的完整内容

数据输入和输出

数据输入: `load`、`fscanf` 和 `input`

数据输出: `save`、`fprintf` 和 `disp`

函数

声明语句: `function [y1, y2, ..., yn] = FuncName(x1, x2, ..., xn)`

检索函数变量数: `nargin()` 和 `nargout()`

函数句柄: `@function_name`

函数值计算: `feval()`

程序流程控制

if 选择语句: `if ... elseif ... else ... end`

switch 结构: `switch ... case ... otherwise ... end`

for 循环结构: `for ... end`

while 循环结构: `while ... end`

try...catch 语句: `try ... catch`

`continue`、`break` 和 `pause` 语句: 常置于 for 循环或 while 循环内, 根据条件执行 `continue`、

break 或 pause 语句。若执行 continue，即跳出循环体中尚未执行的语句，接着进行下一次是否进行循环的判断；若执行到 break 语句，则直接退出最内层的 for 循环或 while 循环；若执行 pause 语句程序将暂停以等待用户按键，当用户按下任意键时程序才继续往下执行。

return 语句：在某个函数内部执行 return 语句时，可立即退出该函数，并返回到调用它的函数，继续运行。

常用内部函数

一些通用计算函数：

max	min	mean	sum	sqrt	exp	log	sort
最大值	最小值	算术平均	求和	平方根	指数	自然对数	排序

其中，sqrt、exp、log 支持数组运算。

矩阵函数：

eye	单位矩阵
ones	元素均为 1 的矩阵
zeros	元素均为 0 的矩阵
rand	随机矩阵
linspace	等间距的行向量
logspace	对数等间距的行向量
sparse	稀疏矩阵
det	求矩阵行列式
eig	特征值、特征向量
inv	逆阵
rank	求矩阵的秩
svd	矩阵的奇异值分解
find	寻找向量或矩阵中符合指定条件的元素，并返回其下标
fsolve	解代数方程组
fzero	解单变量代数方程
impulse	脉冲响应
step	阶跃响应
ode45	积分求解常微分方程组
polyfit	多项式最小二乘拟合
ss2tf	把状态空间转变为传递函数模型
tf2ss	把传递函数转变为状态空间模型

符号运算：

sym	创建符号表达式
syms	快速创建符号对象
vpa	返回符号表达式在 digits 函数设置的精度下的数值解
subs	符号替代函数，如 subs(s, old, new)
diff	微分和差分
gradient	梯度
jacobian	雅可比矩阵

<code>solve</code>	代数方程（组）
<code>dsolve</code>	ODE 方程（组）
绘图函数：	
<code>plot</code>	绘制二维线形图
<code>hold on</code>	保持当前图，允许新图画在当前图上
<code>hold off</code>	关闭保持功能
<code>text</code>	将文本放置于指定位置上
<code>gtext</code>	在图上放置文本，位置由鼠标点击确定
<code>subplot</code>	在一个图形窗口中画多个图
<code>grid on</code>	对当前坐标添加网格线
<code>grid off</code>	关闭网格线
<code>xlabel</code>	在 x 轴下方添加文本
<code>ylabel</code>	在 y 轴左侧添加文本
<code>zlabel</code>	在 z 轴左侧添加文本
<code>title</code>	在图形顶部添加图的标题
<code>axis</code>	设置坐标轴的上下边界（坐标轴范围）
<code>subplot</code>	在不同框架（frames）中显示多个图形：
<code>shg</code>	用 shg 可把当前图形窗口显示出来：
<code>clf</code>	在继续画新的图形时，可用 clf 命令清除已画的图形窗口
<code>figure</code>	可画出两个以上的图形窗口
<code>meshgrid</code>	生成用于绘制三维图形的 x 和 y 数组
<code>mesh</code>	绘制三维网格曲面图
<code>surf</code>	绘制三维彩色曲面图

参 考 文 献

- 1 MATLAB Help, Version 6.5. Release 13. The Mathworks, Inc., 2002
- 2 苏金明, 阮沈勇编著. MATLAB 6.1 实用指南（上册）. 北京：电子工业出版社，2002

第3章 MATLAB在数值分析中的应用

3.1 插值与拟合

插值与拟合在科学研究和工程实际中应用很广泛。下面介绍如何用 MATLAB 来处理插值与拟合问题。

3.1.1 数据插值

常见的插值方法有 Lagrange 多项式插值、牛顿插值、分段线性插值、Hermite 插值和三次样条插值等。MATLAB 主要提供分段线性插值和三次样条插值,如表 3-1 所示。对于 Lagrange 多项式插值、牛顿插值和 Hermite 插值, MATLAB 没有提供相应的函数,若想用这几种插值方法可自己编写函数或直接参考文献[1~3]。

表 3-1 MATLAB 的数据插值 (Data Interpolation) 函数

插值方法	MATLAB 函数	插值方法	MATLAB 函数
一维插值	interp1	使用 FFT 方法的一维插值	interpft
快速一维插值	interp1q	分段三次 Hermite 插值	pchip
二维插值	interp2	三次样条插值	spline
三维插值	interp3	分段多项式估计函数	ppval
N 维插值	interpN		

下面主要介绍一维插值、二维插值、三维插值和三次样条插值。

一维插值: interp1()

其调用格式为:

$y_i = \text{interp1}(x, y, x_i)$ 已知数据向量 (x, y) , 计算并返回在插值向量 x_i 处的函数值向量 y_i 。

$y_i = \text{interp1}(x, y, x_i, \text{'method'})$

$y_i = \text{interp1}(x, y, x_i, \text{'method'}, \text{'extrap'})$

'method'用于指定插值算法,其值可以是:

'nearest'——最近插值

'linear'——线性插值(默认值)

'spline'——分段三次样条插值

'pchip'——分段三次 Hermite 插值

'cubic'——与'pchip'相同

执行以上前两条语句时,若向量 x_i 中有元素不在 x 的范围内,则对应 y_i 值为 NaN,而第三条语句中的'extrap'用于指定当向量 x_i 中有元素不在 x 的范围内时,采用'method'所指定的插值算法进行外插计算与之对应的 y_i 值。三条语句均要求数据向量 x 为单调。若 y 为矩阵,则对 y 的每一列进行插值,返回的矩阵 y_i 的大小为 $\text{length}(x_i) \times \text{size}(y, 2)$ 。

【例 3-1】 用函数 $y = e^x$ 生成下列离散数据。

x	2.5	2.6	2.7	2.8	2.9
y	12.1825	13.4637	14.8797	16.4446	18.1741

试利用这些离散数据插值计算 $x = [2.55 \ 2.63 \ 2.77 \ 2.86]$ 处的函数值，并与真实值比较。

程序说明 分别用最近插值、线性插值、分段三次样条插值和分段三次 Hermite 插值等算法进行计算。

程序清单 *xInterpl.m*

```
clear all, clc
x = [2.5 2.6 2.7 2.8 2.9]; y = exp(x); % 已知离散点(x,y)
xi = [2.55 2.63 2.77 2.86]; % 插值向量

% 一维插值
yi_interp1_nearest = interp1(x, y, xi, 'nearest') % 最近插值
yi_interp1_linear = interp1(x, y, xi) % 线性插值 (默认)
yi_interp1_spline = interp1(x, y, xi, 'spline') % 分段三次样条插值
yi_interp1_pchip = interp1(x, y, xi, 'pchip') % 分段三次 Hermite 插值
y_true = exp(xi) % 对应于插值向量的真实函数值
```

计算结果 在 $x = [2.55 \ 2.63 \ 2.77 \ 2.86]$ 处的函数值如下：

真实值	12.8071	13.8738	15.9586	17.4615
最近插值	13.4637	13.4637	16.4446	18.1741
线性插值	12.8231	13.8885	15.9752	17.4823
三次样条插值	12.8071	13.8738	15.9586	17.4616
分段三次 Hermite 插值	12.8067	13.8737	15.9588	17.4622

可见，三次样条插值算法的插值精度最高。

二维插值: `interp2()`

`interp2` 函数的使用方法与一维线性插值函数 `interp1` 完全相似。调用格式为：

$zi = \text{interp2}(x, y, z, xi, yi, 'method')$

说明 输入参数 (x, y, z) 为已知的离散数据向量，其中 z 为对应于 x, y 处的函数值向量。

'method' 算法属性值可以是：

'nearest' ——最近插值

'linear' ——线性插值 (默认)

'spline' ——三次样条插值(spline)

'cubic' ——立方插值

注 以上要求数据向量 x 和 y 为单调，且格式相同。若向量 xi, yi 中有元素不在 x, y 的范围内，则与之相对的 zi 返回值为 NaN。此外， x, y, z 也可以是矩阵形式。

【例 3-2】 由 $z = e^x \sin y + y - 0.1$ 生成下表的离散数据，试利用这些离散数据插值计算在 $xi = 1.9, yi = 0.36$ 处的 z 值，并与真实值比较。

$x \backslash y$	0.1	0.2	0.3	0.4	0.5	0.6
0.5	0.1646	0.4276	0.6872	0.9420	1.1904	1.4309
1.0	0.2714	0.6400	1.0033	1.3585	1.7032	2.0349
1.5	0.4474	0.9904	1.5244	2.0453	2.5486	3.0306
2.0	0.7377	1.5680	2.3836	3.1774	3.9425	4.6722
2.5	1.2162	2.5203	3.8002	5.0441	6.2406	7.3788
3.0	2.0052	4.0904	6.1357	8.1217	10.0295	11.8411

程序清单 见光盘中的 *xInterp2.m*。

计算结果 在 $x_i = 1.9$ 、 $y_i = 0.36$ 处的函数插值结果为 $z_i = 2.5244$ ，而真实值为 $z_{\text{true}} = 2.6153$ 。

三维插值: interp3()

三维插值函数 `interp3()` 的使用方法与一维和二维线性插值函数 `interp1` 和 `interp2` 完全相似。调用格式为: $v_i = \text{interp3}(x, y, z, v, x_i, y_i, z_i, \text{'method'})$ ，其中输入参数 (x, y, z, v) 为已知的离散数据向量， v 为对应于 x, y, z 处的函数值向量，'method' 算法属性值同前。向量 x, y, z 的长度必须一样。

三次样条插值: spline()

由于三次样条函数能够给出光滑的插值曲线，因此特别得到研究人员和工程师的青睐。

MATLAB 提供 `interp1` 和 `spline` 两个三次样条插值函数，其中 `interp1()` 前面已介绍，将其 'method' 参数值设为 'spline'，`interp1()` 即采用分段三次样条插值方法。`spline()` 调用格式为：

$y_i = \text{spline}(x, y, x_i)$ 此函数等同于 $y_i = \text{interp1}(x, y, x_i, \text{'spline'})$

$pp = \text{spline}(x, y)$ 返回三次样条插值的分段多项式形式的向量，供 `ppval()` 调用以进行插值计算: $y_i = \text{ppval}(pp, x_i)$

注 由于样条曲线应用广泛，MATLAB 专门提供了样条工具箱 (Spline Toolbox)。该工具箱中的一些函数，具有很强的插值与拟合功能。这里介绍的函数 `spline()`，并非是样条工具箱函数，而是 MATLAB 的一个内部函数。但作者认为，`spline()` 在解决科学与工程计算中的常见插值问题已经非常胜任，因此不再介绍样条工具箱中有关插值方面的功能。有关样条工具箱中的拟合函数将在 2.1.2.2 中介绍。

【例 3-3】 对[例 3-1]的问题(程序 *xInterp1.m*)，改用三次样条插值函数 `spline()` 计算。

程序说明: 把 *xInterp1.m* 中的插值语句改为 $y_i_{\text{spline}} = \text{spline}(x, y, x_i)$ 即可，并另存为文件 *xSpline.m*。运行 *xSpline.m*，并与 *xInterp1.m* 的运行结果比较可知，函数 `spline()` 与函数 `interp1()` 在其 'method' 参数值设为 'spline' 时的结果完全一样。

3.1.2 最小二乘法拟合

在科学研究与工程实践中，常常测量得到许多离散的实验数据和工业数据，这些数据通常需要经过各种拟合方法以得到连续光滑的曲线。其中，最常用的是最小二乘曲线拟合。

已知离散实验数据 (x_i, y_i) ，用最小二乘法可以拟合因变量 y 和自变量 x 之间的函数关系 $y = f(x)$ ，其基本思路是，在给定点 x_i 上使残差平方和 $\sum (f(x_i) - y_i)^2$ 最小。由于观测数据往往带有实验误差，而最小二乘拟合方法并不要求拟合函数 $y = f(x)$ 经过所有的实验点 (x_i, y_i) ，只要求残差平方和最小，因此，用最小二乘法来拟合实验数据是非常合适的。

3.1.2.1 最小二乘多项式拟合

利用最小二乘多项式拟合函数 `polyfit()`，可实现最小二乘曲线拟合。其调用格式如下：

$p = \text{polyfit}(x, y, n)$

$[p, s] = \text{polyfit}(x, y, n)$

输入参数: (x, y) 为已知数据向量， n 为多项式阶数；输出参数 p 为拟合生成的多项式的系数向量（长度为 $n+1$ ）， s 为结构参数，供函数 `polyval()` 调用以获得误差估计值。

函数 `polyval()` 常常与 `polyfit()` 联合使用，其调用格式为: $y = \text{polyval}(p, x)$ 。

`polyval()` 根据由 `polyfit()` 算得的多项式系数向量 p ，在 x 处估计相应的多项式函数值 y ：

$$y = p(1) \cdot x^n + p(2) \cdot x^{n-1} + \cdots + p(n) \cdot x + p(n+1) \quad (3-1)$$

注 ①多项式 y 中自变量的次方从高到低排列, 即 x^n, x^{n-1}, \dots, x^1 , 而其系数分别是 $p(1), p(2), \dots, p(n)$, 最后一项为 $p(n+1)$; ②polyfit()仅处理单变量多项式的最小二乘拟合问题。③线性回归是 $n=1$ 时的特例。

【例 3-4】 给出如下数据表

i	0	1	2	3	4
t_i	0	0.25	0.50	0.75	1.00
y_i	1.093	1.2560	1.6387	2.1570	2.7183

求最小二乘拟合二次多项式。

程序清单 *xPolyfit.m*

```
clear all; clc
t = [0 0.25 0.50 0.75 1.00];
y = [1.093 1.2560 1.6387 2.1570 2.7183];
p = polyfit(t, y, 2)
ti = linspace(t(1), t(end), 100);
yi = polyval(p, ti);
plot(t, y, 'o', ti, yi, '-'), xlabel('t'), ylabel('y')
```

计算结果 拟合的二次多项式为: $p_2(x) = 1.0654x^2 + 0.5953x + 1.0755$, 拟合曲线如图 3-1。

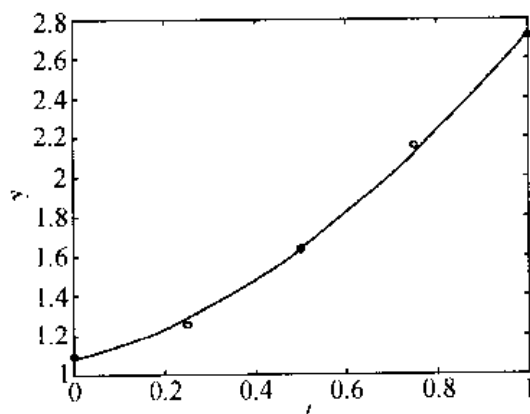


图 3-1 最小二乘拟合多项式拟合

对于简单的非线性模型, 可先进行线性化, 然后用线性最小二乘法拟合, 现举例如下。

【例 3-5】 给出数据如下表。

i	1	2	3	4	5
x_i	1.00	1.25	1.50	1.75	2.00
y_i	5.10	5.79	6.53	7.45	8.46

求形如 $y = ae^{bx}$ 最小二乘拟合函数。

程序说明 对 $y = ae^{bx}$ 两边取对数, 得 $\ln y = \ln a + bx$, 故程序可用 polyfit() 进行线性拟合。

程序清单 *xPolyfitLinear.m*

```
clear all; clc
xi = [1.00 1.25 1.50 1.75 2.00];
yi = [5.10 5.79 6.53 7.45 8.46];
Lnyi = log(yi);
p = polyfit(xi, Lnyi, 1)           % “1”指定线性拟合，拟合：lny = bx + lna
b = p(1), Lna = p(2); a = exp(Lna)
x = linspace(xi(1), xi(end), 100);
Lny = polyval(p, x)
plot(xi, Lnyi, 'o', x, Lny, '-'), xlabel('x'), ylabel('lny')
```

计算结果 $a = 3.073$, $b = 0.5057$ 即 $y = 3.073e^{0.5057x}$

3.1.2.2 用最小二乘法拟合生成样条曲线

与样条插值不同，样条拟合并不要求曲线通过全部的数据点。在工程实践与科学研究中，由实验观测得到的一组离散数据一般包含实验误差，因此，样条拟合比样条插值应用更广泛。

MATLAB 样条工具箱提供的函数较多（执行命令 `>>help splines` 可快速查看函数列表），最常用的三个拟合函数表 3-2 所示。

表 3-2 常用的三个拟合函数

函数名	曲线类型	拟合准则	是否平滑处理
csaps()	三次样条曲线	最小二乘法	是
spap2()	B 样条曲线	最小二乘法	否
spaps()	B 样条曲线	最小二乘法	是

使用这些函数拟合得到曲线函数 *sp* 以后，可利用 *fnval()* 计算任意自变量下的函数值。下面介绍这几个函数的用法。

函数 *csaps()* 的用法

功能：平滑生成三次样条函数，即对于数据 (x_i, y_i) ，所求的三次样条函数 $y = f(x)$ 满足

$$\min_p \sum_i w_i (y_i - f(x_i))^2 + (1-p) \int \lambda(t) (D^2 f)^2 dt \quad (3-2)$$

调用格式：`sp = csaps(x, y, p)`

`ys = csaps(x, y, p, xx, w)`

输入参数：`x, y` 要处理的离散数据 (x_i, y_i)

`p` 平滑参数，取值区间为 $[0, 1]$ 。当 $p=0$ 时，相当于最小二乘直线拟合；当 $p=1$ 时，相当于“自然的”三次样条插值，即相当于 *csapi()* 或 *spline()*。

`xx` 用于指定在给定点 `xx` 上计算其三次样条函数值 (`ys`)

`w` 权值（权重），默认为 1

输出参数：`sp` 拟合得到的样条函数

`ys` 在给定点 `xx` 上的三次样条函数值

函数 *spap2()* 的用法

功能：用最小二乘法拟合生成 B 样条曲线，即：

对于离散数据 (x_i, y_i) , 所求的 k 次样条函数 $y = f(x)$ 满足

$$\min \sum_i w_i (y_i - f(x_i))^2 \quad (3-3)$$

调用格式: `sp = spap2(knots, k, x, y)`

`sp = spap2(knots, k, x, y, w)`

输入参数: `knots` 节点序数 (knot sequence)

`k` 样条函数的阶次, 一般取 $k = 3$, 有时取 $k = 4$

`x, y` 要处理的离散数据 (x_i, y_i)

`w` 权值 (权重), 默认为 1

输出参数: `sp` 拟合得到的样条函数

函数 `spaps()` 的用法

功能: 平滑生成 B 样条函数, 所用最小二乘法拟合准则为 (3-3) 式 (同函数 `spap2`)。

调用格式: `sp = spaps(x, y, tol)`

`[sp, ys] = spaps(x, y, tol, m, w)`

输入参数: `x, y` 要处理的离散数据 (x_i, y_i)

`tol` 光滑时的允许精度

`m` 默认值是 2, 即平滑生成三次 B 样条曲线

`w` 权值 (权重), 默认为 1

输出参数: `sp` 拟合得到的样条函数

`ys` 在 x 上经平滑处理的 B 样条函数值

函数 `fnval()` 的用法

功能: 计算函数 f 在给定 x 处的函数值 `values`。

调用格式: `values = fnval(f, x)`

【例 3-6】 先生成离散数据 (x, y) , 其中, x 为四个周期内的离散数据, $y = \cos(x) + \varepsilon$, ε 为随机误差, 然后分别用函数 `csaps()`、`spap2()` 和 `spaps()` 拟合这些离散数据并作图比较。

程序清单 `xSplineFit.m`

```
% 生成离散数据(x,y), 其中, x 为四个周期, y 为加入随机误差的余弦波
x = linspace(0, 4*pi, 41); y = cos(x) + (rand(1, 41) - 0.5)*0.2;
pp = csaps(x, y, 0.98); % By csaps()
knots = 9; K = 4;
sp1 = spap2(knots, K, x, y); % By spap2()
sp2 = spaps(x, y, 0.016); % By spaps()

% 分别绘制三种方法的拟合曲线
plot(x, y, 'ok'); hold on; fnplt(pp, 'r'); hold off % By csaps()
figure, plot(x, y, 'ok'); hold on; fnplt(sp1, 'b'); hold off % By spap2()
figure, plot(x, y, 'ok'); hold on; fnplt(sp2, 'r'); hold off % By spaps()

% 在同一坐标上绘制三条曲线以便比较
figure, plot(x, y, 'ok'); hold on, fnplt(pp, 'k'), fnplt(sp1, 'b'), fnplt(sp2, 'r')
xlabel('x'), ylabel('y'), hold off
```

计算结果 如图 3-2 所示, 由三条函数拟合所得的曲线基本重合在一起, 拟合效果均较好。显然, 能对余弦波拟合得好, 说明其拟合能力较强。

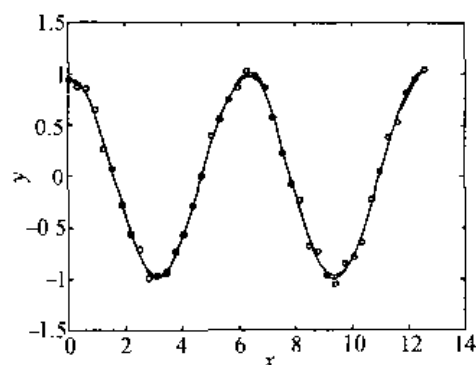


图 3-2 样条拟合曲线

3.1.3 应用实例

【例 3-7】 青霉素发酵的实验数据如表 3-3 所示, 请用插值方法估算 $t = 10, 30, 50, 70, 90, 110, 130, 150, 170, 190$ h 时青霉素的浓度。

表 3-3 青霉素发酵的实验数据(data of penicillium fermentation)^[3]

t/h	青霉素浓度 (单位/ml) ^①	t/h	青霉素浓度 (单位/ml) ^①
0	0	120	9430
20	106	140	10950
40	1600	160	10280
60	3000	180	9620
80	5810	200	9400
100	8600		

① 青霉素浓度单位一般是“生物效价单位/ml”(简称“单位/ml”), 通过一定的转换系数可将其换为“g/ml”。

程序说明 将表 3-3 的实验数据存于文件 FermData.m 中。程序先调用此文件获得实验数据, 然后用 spline() 进行三次样条插值计算。

程序清单 FermInterp.m

```
clear all; clc

% 调入青霉素发酵的实验数据
FermData
t = FermData(:, 1); C = FermData(:, 2); % t: h; C: 青霉素浓度 (单位/ml)

% 求插值点 ti=[10, 30, 50, 70, 90, 110, 130, 150, 170, 190]处的青霉素浓度 (单位/ml)
ti = 10:20:190;
Ci = spline(t, C, ti)

% 插值计算间隔很小的离散数据, 以便绘制光滑的样条插值曲线
pp = spline(t, C); tj = 0:0.5:200; Cj = ppval(pp, tj);

% 图形输出
plot(t, C, 'ko', ti, Ci, 'b*', tj, Cj, 'b-'), legend('实验点', '插值点', '样条插值曲线')
xlabel('时间 (h)'), ylabel('青霉素浓度 (单位/ml)')
```


计算结果 在插值点 $t_i = [10, 30, 50, 70, 90, 110, 130, 150, 170, 190]$ 处的青霉素浓度 (单位/ml) 为: $C_i = 10^4 \times [-0.0274 \ 0.0833 \ 0.2223 \ 0.4240 \ 0.7420 \ 0.9061 \ 1.0267 \ 1.0822 \ 0.9861 \ 0.9489]$ 。第一个数值为负, 不符合实际物理意义, 这是由插值误差造成的。

为直观考察插值性能, 程序还绘出图形, 如图 3-3。由图可见, 插值结果是比较满意的。

注 此例仅演示插值方法, 对于表 3-3 中的实验数据, 因存在实验误差, 用拟合应更符合实际 (参见例 3-8)。

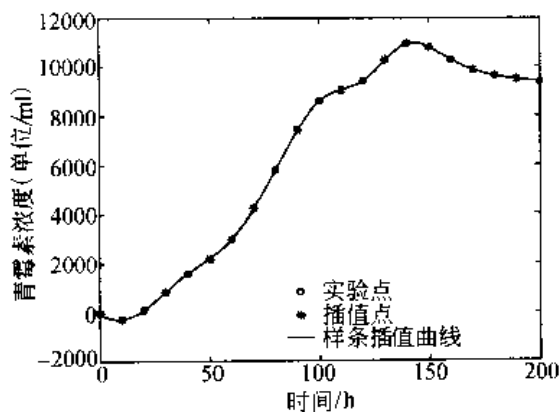


图 3-3 三次样条插值

【例 3-8】 青霉素发酵的实验数据同[例 3-7]的表 3-3, 请用最小二乘法进行拟合, 并估算 $t = 10, 30, 50, 70, 90, 110, 130, 150, 170, 190$ h 时青霉素的浓度。

程序说明 程序使用最小二乘样条拟合和最小二乘多项式拟合两种方法, 以便比较。

程序清单 见光盘中的 *FermDataFit.m*。

计算结果

(1) 当程序中样条拟合参数 $knots = 2$, $K = 4$, 多项式拟合阶次 $m = 4$ 时, 用两种方法计算得到 $t_i = [10, 30, 50, 70, 90, 110, 130, 150, 170, 190]$ 处的青霉素浓度基本一样:

$Ci1 = [-199 \ 433 \ 2290 \ 4759 \ 7226 \ 9093 \ 10129 \ 10466 \ 10251 \ 9633]$ (样条拟合)

$Ci2 = [-267 \ 410 \ 2354 \ 4801 \ 7164 \ 9031 \ 10172 \ 10530 \ 10228 \ 9565]$ (多项式拟合)

与[例 3-7]的插值类似, 第一个数值为负, 不符合实际物理意义, 从拟合曲线图 3-4(a)也可以看出, 第一和第二个散点之间的一段曲线为负, 因此, 拟合精度仍有待进一步提高。

(2) 当程序中 $knots = 5$, $K = 4$, $m = 6$ 时, 用两种方法计算得到 t_i 处的青霉素浓度基本一样, 拟合曲线如图 3-4(b), 由图可见, 拟合曲线基本重叠, 结果基本满意。但严格地说, 最后两个散点之间的一段曲线, 存在青霉素浓度先下降后上升的趋势 (由数组 $C4plot2$ 也可观察到), 这与实际情况不符, 因此, 仍可以通过微调拟合参数得以改善。

(3) 通过改变多项式阶次的计算结果发现, 最小二乘多项式拟合已难再进一步改善 (阶次改为 7 以上时又出现负值, 结果从略), 而最小二乘样条拟合仍可改善。当程序中 $knots = 6$, $K = 4$, $m = 6$ 时, 在 t_i 处的青霉素浓度计算结果为:

$Ci1 = [105 \ 691 \ 2051 \ 4566 \ 7261 \ 8988 \ 10421 \ 10718 \ 9880 \ 9491]$ (样条拟合)

$Ci2 = [21 \ 644 \ 2220 \ 4533 \ 7053 \ 9176 \ 10425 \ 10618 \ 10000 \ 9344]$ (多项式拟合)

拟合曲线如图 3-4(c)。从变化的趋势看, 图 3-4(c)的拟合曲线应是比较满意的。

由此可见, 最小二乘样条拟合函数 `spap2()` 的拟合效果优于最小二乘多项式拟合函数 `polyfit()`。从两者的可调参数看, 前者有 `knots` 和 `K` 两个可调参数, 后者只有一个可调参数(阶次), 因此, 前者更灵活, 这也是由 B 样条本身的性质所决定的。

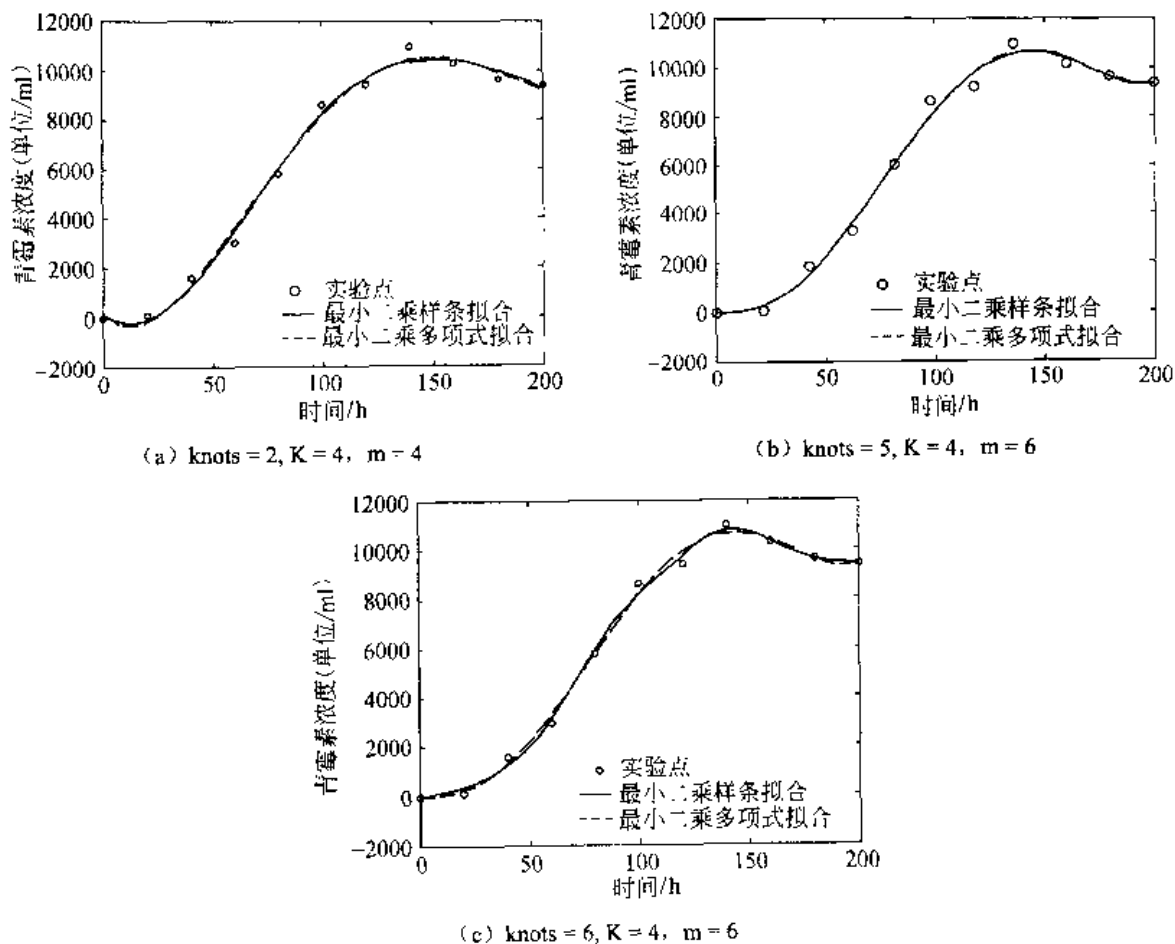


图 3-4 青霉素发酵数据的最小二乘拟合

3.2 数值积分与数值微分

许多工程问题的求解需要计算函数在某个已知点的导数, 或者计算导数在自变量已知范围内的积分。尽管在一些场合, 可直接对函数求导, 但如果函数复杂以及/或者需要由计算机程序计算的话, 那么数值计算将更加方便。对于实验数据, 常常缺乏相应的代数表达式, 或者函数不能积分得到分析解, 因此数值微分或数值积分是必不可少的。

3.2.1 数值积分

数值积分 (Numerical Quadrature) 方法主要有 Newton-Cotes 系列数值积分法、Gauss 积分法和 Romberg 积分法等。MATLAB 只提供 Newton-Cotes 系列数值积分法的有关计算函数, Gauss 积分法和 Romberg 积分法的有关计算函数可参考文献[1~3]。

积分形式: 已知函数 $y=f(x)$, 在区间 $[a, b]$ 内 y 对 x 的积分为
$$I = \int_a^b y dx \quad (3-4)$$

几种常用的 Newton-Cotes 系列数值求积公式及其相应的 MATLAB 函数, 如表 3-4 所示。

表 3-4 常用的 Newton-Cotes 公式及其 MATLAB 函数

n	公 式	MATLAB 函数	精 度
1	梯形求积公式	trapz	低 ↓ 高
2	自适应 Simpson 求积公式 (低阶)	quad	
3	自适应 Lobatto 求积公式 (高阶)	quadl	

下面介绍以上几个 MATLAB 求积函数的调用方法。

梯形法数值积分: trapz()

$z = \text{trapz}(y)$ 用梯形求积方法计算 y 的积分近似值。对于向量 y , $\text{trapz}(y)$ 返回 y 的积分; 对于矩阵 y , $\text{trapz}(y)$ 返回一行向量, 向量中的各元素为矩阵 y 的对应列向量的积分值; 对于 N 维数组, $\text{trapz}(y)$ 从第一个非独立维开始计算。

自适应 Simpson 法数值积分: quad()

调用格式: $q = \text{quad}(@\text{fun}, a, b)$

$q = \text{quad}(@\text{fun}, a, b, \text{tol})$

$q = \text{quad}(@\text{fun}, a, b, \text{tol}, \text{trace}, p1, p2, \dots)$

输入参数: fun 被积函数。在定义 fun 时, 被积函数表达式必须是向量形式, 即表达式必须使用点运算符 ($.*$ 、 $./$ 和 $.^$) 以支持向量

a, b 即积分限 $[a, b]$

tol 绝对误差限, 默认值为 $1.e-6$

$p1, p2, \dots$ 直接传递给函数 fun 的已知参数

输出参数: q 积分结果

自适应 Lobatto 法数值积分: quadl()

函数 $\text{quadl}()$ 比 $\text{quad}()$ 更有效, 精度更高。其使用方法与函数 $\text{quad}()$ 完全相同。

【例 3-9】对 $I = \int_0^{3\pi} \exp(-0.5t) \sin(t + \pi/6) dt$ 进行数值积分。

程序说明 用梯形求积函数 $\text{trapz}()$ 、自适应 Simpson 求积 (低阶) 函数 $\text{quad}()$ 和自适应 Lobatto 求积 (高阶) 函数 $\text{quadl}()$ 进行计算。

注 被积函数 $\text{func}()$ 的表达式必须是向量形式: $y = \exp(-0.5*t) .* \sin(t+\pi/6)$, 式中 t 和 y 为向量, 所以用到点乘符号 “ $.*$ ”。若为非向量形式: $y = \exp(-0.5*t) * \sin(t+\pi/6)$, 则用 $\text{quad}()$ 和 $\text{quadl}()$ 求积时将出错。

程序清单 xIntegration.m

```
function xIntegration
clear all; clc
a = 0; b = 3*pi; d = pi/1000; % a、b 和 d 分别是积分下限、积分上限和积分步长
t = a:d:b; y = func(t); format long
y_trapz = trapz(y) * d % 梯形数值积分(Trapezoidal numerical integration)
y_quad = quad(@func, a, b) % 自适应 Simpson 法(Adaptive Simpson quadrature, low
order)
y_quadl = quadl(@func, a, b) % 自适应 Lobatto 法(Adaptive Lobatto quadrature, high
order)

disp('Results:')
```

```

disp('   Trapz           Qquad           Quadl')
disp([y_trapz,y_quad,y_quadl])

% -----
function y = func(t)
y = exp(-0.5*t) .* sin(t+pi/6);

```

计算结果

Trapz	Quad	Quadl
0.90084027660688	0.90084081100646	0.90084078775646

奇异积分：奇异积分分为反常积分（被积函数在某一点上无界）与无穷积分。

【例 3-10】 计算 $I = \int_0^1 \frac{dx}{x^{1/2} + x^{1/3}}$

程序说明 由于在 $x=0$ 处，被积函数为无穷大，因此 I 是奇异积分。若用 `trapz()` 进行此奇异积分，结果将是 `Inf`（无穷大），显然不正确，所以该函数不适用，读者感兴趣可以试试。这里使用自适应 Simpson 求积函数 `quad()` 和自适应 Lobatto 求积函数 `quadl()` 进行计算。

程序清单 见光盘中的 `xSingularInt.m`。

计算结果 用 `quadl()` 求得 `y_quadl = 0.84111708263095`，用 `quad()` 求得 `y_quad = 0.84113024315845`，而准确值为：0.84111692。显然，自适应高阶求积函数 `quadl` 得到的结果更接近准确值。所以，建议优先采用自适应高阶求积函数 `quadl()`。

以上例 3-9 和例 3-10 都是对已知函数在给定区间的数值积分，而实际应用中将涉及离散数据（表格数据）的数值积分，在 3.2.3 节中将给出这方面的例子。

3.2.2 数值微分

在科学研究与工程实际中，常常需要根据一组离散数据点，计算函数在某一点的导数。数值微分就是解决这类问题的方法，其基本思路是：先用拟合方法（最小二乘法）根据已知数据拟合得近似函数，再对此近似函数进行微分，然后对微分后得到的多项式求值，即可求出在拟合范围内任意一点处的任意阶微分。但是，对拟合多项式求高阶导数时，阶数越高计算误差就越大。因此，通常仅限于低阶数值微分。

由离散数据求数值微分的四种方法及有关 MATLAB 函数。

(1) **有限差分法** 用差分函数 `diff()` 近似计算导数，即 $dy = \text{diff}(y)/\text{diff}(x)$ 。此方法用一阶差分，精度较差，若改用二阶差分，可大大提高精度，但编程麻烦些。

(2) **多项式拟合法** 离散数据 $\xrightarrow{\text{polyfit()}}$ 向量 `p` 表示的多项式拟合函数 $\xrightarrow{\text{polyder()}}$ 导函数 `pp` $\xrightarrow{\text{polyval()}}$ `pp` 在 `xi` 的导数值。其中，函数 `polyfit()` 和 `polyval()` 已在 3.1.2 节中介绍，而求导函数 `polyder()` 的调用格式为：

`pp = polyder(p)` % 对向量 `p` 表示的多项式函数进行求导，返回导函数为 `pp`

(3) **三次样条插值方法** 离散数据 $\xrightarrow{\text{csapi()}}$ 三次样条插值函数 `cs` $\xrightarrow{\text{fnder()}}$ `cs` 的导数 `pp` $\xrightarrow{\text{fnval()}}$ `pp` 在 `xi` 的导数值。由于插值曲线经过离散点，而最小二乘拟合不一定经过离散点，因此从实际情况考虑，使用这种插值方法一般不如使用最小二乘拟合方法 (2)、(4) 好。

(4) **样条拟合法（最小二乘法）** 离散数据 $\xrightarrow{\text{csaps() 或 spap2() 或 spaps()}}$ 样条拟合函数 `sp` $\xrightarrow{\text{fnder()}}$ `sp` 的导数 `pp` $\xrightarrow{\text{fnval()}}$ `pp` 在 `xi` 的导数值。其中，函数 `csaps()`、`spap2()`、`spaps()` 和 `fnval()` 已在 3.1.2 节中介绍，而求导函数 `fnder()` 的调用格式为：

pp = fnder(f, dorder) % 计算函数 f 的 dorder 阶导函数, 默认 dorder = 1 (1 阶)

上述四种方法中, 方法 (2) 和方法 (4) 较好。

【例 3-11】 已知函数 $y = \cos(x)$, 计算在 $x = [0: \pi/20: \pi]$ 下的对应函数值向量 y , 试根据这些离散数据 (x, y) 计算 $x = [0: \pi/20: \pi]$ 处函数的一阶导数值。

程序说明 用有限差分法、多项式拟合方法和三次样条插值方法对离散数据进行数值微分:

(1) 有限差分法: 用差分函数 `diff()` 近似计算导数, 即 $dy = \text{diff}(y)/\text{diff}(x)$;

(2) 多项式拟合方法: 先用 `polyfit()` 根据离散数据拟合得到多项式插值函数 p , 再用 `polyder()` 计算 p 的导数 pp , 然后用 `polyval()` 计算 pp 在 x 的导数值;

(3) 三次样条插值方法: 先用 `csapi()` 根据离散数据生成三次样条插值函数 cs , 再用 `fnder()` 计算 cs 的导数 pp , 然后用 `fnval()` 计算 pp 在 xi 的导数值。

程序清单 xDifferential.m

```
function xDifferential
clear all; clc

% 用函数 y = cos(x) 生成离散数据及 x 处的导数准确值
x = [0:pi/20:pi]; y = cos(x);
dy = -sin(x) % 导数准确值

% 有限差分法
dy1 = diff(y)/diff(x), x1 = x(1:length(x)-1);

% 多项式拟合方法
p = polyfit(x, y, 5); % 进行多项式拟合, 得到多项式 p
pp = polyder(p); % 对拟合得到的多项式求导, 得到导函数 pp
dy2 = polyval(pp, x) % 计算导函数 pp 在 x 的导数值

% 三次样条插值方法
cs = csapi(x, y); % 生成三次样条插值函数 cs
pp = fnder(cs); % 计算三次样条插值函数 cs 的导函数 pp
dy3 = fnval(pp, x) % 计算导函数 pp 在 x 的导数值

% 多项式拟合方法和三次样条插值方法的结果与准确值之差
disp('多项式拟合方法和三次样条插值方法的结果与准确值之差[dy-dy2;dy-dy3]: ')
disp([dy-dy2; dy-dy3])

% 绘制几种方法的导函数曲线
plot(x, dy, 'ko', x1, dy1, 'g-', x, dy2, 'b-', x, dy3, 'r--'), xlabel('x'), ylabel('y')
title('函数 cos(x) 的导函数曲线')
legend('准确值', '有限差分法', '多项式拟合方法', '三次样条插值方法')
```

计算结果 函数 $y = \cos(x)$ 在 $x = [0: \pi/20: \pi]$ 处的一阶导数值可在程序运行后在命令窗口中观察到 (dy 为准确值, $dy2$ 和 $dy3$ 分别是多项式拟合求导方法和三次样条插值求导方法的结果), 这里从略。计算结果示于图 3-5 中。可见, 多项式拟合求导方法 (阶数为 5) 和三次

样条插值求导方法所得曲线重合，结果都很满意，而有限差分法得到的曲线很差，这是因为所用的差分公式只有 1 阶精度，若改用中心差分格式（2 阶精度）计算将大大改善精度，但编程比较麻烦。多项式拟合求导方法必须选用合适的阶数，否则插值效果会很差，如阶数取 3 时效果较差。因此需要尝试不同的阶数，这里阶数取 5 是经过尝试后确定的。

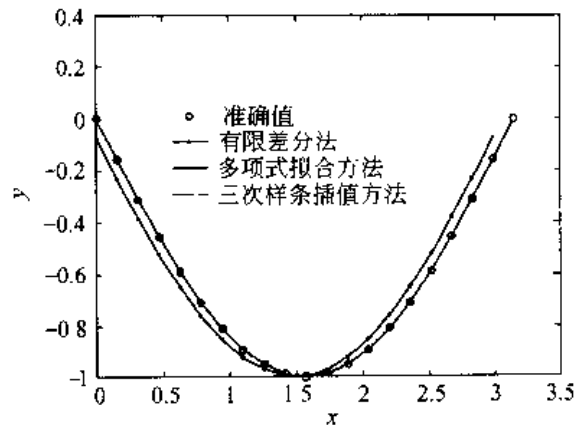


图 3-5 函数 $\cos(x)$ 的导函数曲线

注 这里由于离散数据用 $y = \cos(x)$ 生成，不包括噪声成分，因而用三次样条插值法得到很满意的结果，但对于实际科学研究与工程应用，由于实验数据一般包含噪声（误差），必须改用三次样条拟合方法。

3.2.3 化工应用实例

3.2.3.1 常见的数值积分例子

(1) 双组分简单精馏塔理论板数的计算（Lewis 法）

$$\text{精馏段理论板数} \quad N = \int_{x_f}^{x_d} \frac{dx}{y - x - (x_d - y)/R} \quad (3-5)$$

$$\text{提馏段理论板数} \quad M = \int_{x_w}^{x_f} \frac{dx}{y - x - (y - x_w)/R'} \quad (3-6)$$

在工程计算中，汽液相平衡关系常常是一组离散实验数据，因此以上两式需用数值积分方法求解。

(2) 填料吸收塔的填料高度计算

$$\text{基于气相的传质单元数 (NTU) 为} \quad N_{OG} = \int_{y_2}^{y_1} \frac{dy}{(1-y)^2(y-y^*)} \quad (3-7)$$

(3) 理想间歇反应器、管式反应器、循环反应器（计算反应时间或空时、管长等）

间歇反应器（Batch Reactor, BR）

对于恒密度系统（ $\varepsilon_A = 0$ ）

$$t = C_{A0} \int_0^{x_{Af}} \frac{1}{(-r_A)} dx_A = - \int_{C_{A0}}^{C_{Af}} \frac{1}{(-r_A)} dx_A \quad (3-8)$$

对于反应物体积随转化率成正比的变密度系统

$$t = n_{A0} \int_0^{x_{Af}} \frac{1}{(-r_A)V_0(1 + \varepsilon_A x_A)} dx_A = C_{A0} \int_0^{x_{Af}} \frac{1}{(-r_A)(1 + \varepsilon_A x_A)} dx_A \quad (3-9)$$

稳态操作的管式反应器 (Plug Flow Reactor, PFR):

$$\frac{V}{F_{A0}} = \frac{\tau}{C_{A0}} = \int_0^{x_{Af}} \frac{1}{(-r_A)} dx_A \quad (\text{任意 } \varepsilon_A) \quad (3-10)$$

或

$$\tau = \frac{V}{V_0} = \frac{VC_{A0}}{F_{A0}} = C_{A0} \int_0^{x_{Af}} \frac{1}{(-r_A)} dx_A$$

【例】非等温反应器体积的计算

在一个绝热管式反应器中发生液相不可逆反应: $A \rightarrow B$, 纯组分 A 进入反应器, 在反应器出口处测定其浓度, 即可计算出转化率 x 。反应器体积为

$$V = \frac{V_{0A}}{k_1} \int_0^{x_1} \frac{dx}{(1-x) \exp \left[\frac{E_a}{R} \left(\frac{1}{T_1} - \frac{1}{T} \right) \right]} \quad (3-11)$$

式中, V 为反应器体积; V_{0A} 为进口处组分 A 的体积流量; k_1 为温度 T_1 下的反应速率常数; E 为反应活化能; R 为理想气体常数; T 为转化率为 x 时的反应器温度; T_1 是参考温度。

为了积分, 必须通过热量衡算使 x 与温度 T 相互关联起来。对于绝热管式反应器, 假设组分 A 和组分 B 的热容恒为常数, T 由下式计算:

$$T = T_0 + \frac{x(-\Delta H_R)}{c_{pA} + x(c_{pB} - c_{pA})} \quad (3-12)$$

式中, ΔH_R 为反应热; c_{pA} 和 c_{pB} 分别是 A 和 B 的比热容; T_0 为参考温度。

循环反应器

$$\frac{V}{F_{A0}} = (R+1) \int_{\left(\frac{R}{R+1}\right)x_{Af}}^{x_{Af}} \frac{1}{(-r_A)} dx_A \quad (\text{任意 } \varepsilon_A) \quad (3-13)$$

以上几种反应器的计算公式, 积分项往往需要采用数值积分法计算, 尤其是 $\varepsilon_A \neq 0$ 的情况。

(4) 纯气体的逸度系数

$$\ln \phi = \int_0^p \frac{z-1}{p} dp \quad (3-14)$$

(5) 反应器停留时间分布的混合特性

平均停留时间 t_m

$$t_m = \frac{\int_0^\infty C t dt}{\int_0^\infty C dt} \quad (3-15a)$$

方差 σ^2

$$\sigma^2 = \frac{\int_0^\infty C t^2 dt}{\int_0^\infty C dt} - t_m^2 \quad (3-15b)$$

扩散特征数 $D_L/(uL)$ 为

$$\left(\frac{D_L}{uL} \right) = \frac{\sigma^2}{2t_m^2} \quad (3-15c)$$

3.2.3.2 一些常见的数循微分例子

(1) 干燥速率的实验测定

在干燥过程中, 不同时间间隔记录湿物料的重量, 则干燥速率 R 为

$$R = \frac{1}{A} \frac{dW}{dt} \quad (3-16)$$

式中, W 为湿物料重量; t 为时间; A 为干燥暴露的表面积。

(2) 化学反应动力学实验测定

在化学反应动力学中, 确定反应级数的方法之一是初始速率法。在此方法中, 改变反应

物 A 的初始浓度, 测量 A 的浓度随时间的变化。对每一个初始浓度, 根据反应开始时浓度对时间的微分就可以算出初始反应速率:

$$-r_{A0} = \left. \frac{dC_A}{dt} \right|_{t=0} \quad (3-17)$$

如果反应速率可表示为

$$-r_A = kC_A^n$$

则在 $t=0$ 时,

$$-r_{A0} = kC_{A0}^n$$

两边取对数, 得

$$\ln(-r_{A0}) = \ln k + n \ln C_{A0}$$

由此, 通过线性回归可得到动力学参数 k 和 n 的估计值。当然, 最好以此为初值, 再利用非线性回归方法进行估计 (见第 7 章)。

下面列举数值积分[例 3-12]~[例 3-16]和数值微分[例 3-17]方面的化工应用例子。

【例 3-12】 计算双组分简单精馏塔的理论板数^[5]

氯仿-苯双组分精馏系统的汽液平衡数据如下表所示。对氯仿而言, 规定 $x_f = 0.4$, $x_d = 0.9$, $x_w = 0.15$, 精馏段和提馏段的回流比分别为 $R = 5$, $R' = 4$, 试计算所需的理论板数。

x	0.178	0.275	0.372	0.456	0.650	0.844
y	0.243	0.382	0.518	0.616	0.795	0.931

$$\text{模型 精馏段理论板数} \quad N = \int_{x_f}^{x_d} \frac{dx}{y - x - (x_d - y)/R} \quad (1)$$

$$\text{提馏段理论板数} \quad M = \int_{x_w}^{x_i} \frac{dx}{y - x - (y - x_w)/R'} \quad (2)$$

程序说明

因式 (1) 和式 (2) 的被积函数中 y 与 x 的函数关系以表格 (离散数据) 形式给定, 故需采用拟合法将离散数据 (x_i, y_i) 拟合成多项式, 再将此多项式代入被积函数, 然后求积。

程序中, 函数 func1() 和 func2() 分别定义式 (1) 和式 (2) 的被积函数, 这两个函数都先使用 csaps() 对已知离散数据进行三次样条拟合, 得到拟合函数 sp, 即确定了 $y \sim x$ 函数关系, 这样被积函数也就确定。主程序采用 MATLAB 的自适应 Lobatto 求积函数 quadl()。

运行程序可观察到, 给定的离散数据似乎已经平滑处理过, 用 $sp = csaps(x_i, y_i, 1)$ 拟合效果满意, 这里平滑度取 1, 与三次样条插值命令 “ $sp = spline(x_i, y_i)$ ” 等效。也就是, 对于已经平滑过或者已消除噪音的离散数据, 可用插值处理, 读者可改用 $sp = spline(x_i, y_i)$ 验证。

程序清单 DistStagesCal.m

```
function DistStagesCal % 计算双组分简单精馏塔的理论板数
clear all; clc
xi = [0.178 0.275 0.372 0.456 0.650 0.844]; % 液相平衡浓度
yi = [0.243 0.382 0.518 0.616 0.795 0.931]; % 汽相平衡浓度
xf = 0.4; xd = 0.9; xw = 0.15; % xf、xd 和 xw 分别是进料、塔顶和塔底组成
R = 5; R1 = 4; % R 和 R1 分别是精馏段和提馏段回流比
sp = csaps(xi, yi, 1); % 与 sp = spline(xi, yi) 等效

% 画拟合曲线, 直观地检查拟合效果是否良好
x4plot = linspace(xi(1), xi(end), 200);
```



```

y4plot = fnval(sp, x4plot);
plot(xi, yi, 'o', x4plot, y4plot, '-')

N = quadl(@func1, xf, xd, [], [], sp, xd, R);      % 精馏段理论板数 N
M = quadl(@func2, xw, xf, [], [], sp, xw, R1);    % 提馏段理论板数 M
N = round(N+0.5); M = round(M+0.5);              % 圆整

disp('计算结果:')
fprintf('\n  精馏段理论板数 N 为: %d %s\n', N, ' (块) ')
fprintf('  提馏段理论板数 M 为: %d %s\n', M, ' (块) ')
fprintf('  共需理论板数为: %d %s\n', N+M, ' (块) ')

% -----
function f = func1(x, sp, xd, R)                  % 定义被积函数(精馏段)
y = fuval(sp, x);
f = 1./(y-x-(xd-y)/R);

% -----
function f = func2(x, sp, xw, R1)                 % 定义被积函数(提馏段)
y = fnval(sp, x);
f = 1./(y-x-(y-xw)/R1);

```

计算结果 精馏段和提馏段的理论板数分别为 5 块和 6 块, 理论板总数为 11 块。

符号说明

M	提馏段理论板数	x_d	塔顶组成, 0.9
N	精馏段理论板数	x_f	进料组成, 0.4
R	精馏段回流比, 5	x_w	塔底组成, 0.15
R_1	提馏段回流比, 4	y	平衡气相组成
x	平衡液相组成		

【例 3-13】 等温管式反应器的管长计算^[5]

某气相二聚反应 $2A \rightarrow S$ 在等温管式反应器内进行, 反应条件为 $T = 638^\circ\text{C}$, $p = 1 \text{ atm}$ ($1 \text{ atm} = 101.3 \text{ kPa}$), 反应器内径 $d = 10 \text{ cm}$ 。已知反应速率为: $(-r_A) = 116 \left(p_A^2 - \frac{1}{1.27} p_S \right)$ (1)

为了控制反应, 向反应物 A 中加入一定惰性气体 I, 两者的摩尔比为 A:I=1:0.5。进气速率 F_0 为 10 kmol/h 。

- ① 试计算转化率为 0.1, 0.2, 0.3, 0.4, 0.5 时所需的反应器管长, 并计算平衡转化率 x_{Ae} 。
- ② 假设现有一长度为 5m 的管式反应器, 试计算相同条件下该反应器的出口转化率。

模型

$$\text{等温管式反应器模型为} \quad V_R = F_{A0} \int_0^{x_A} \frac{1}{(-r_A)} dx = y_{A0} F_0 \int_0^{x_A} \frac{1}{(-r_A)} dx \quad (2)$$

$$\text{因此, 反应管长为} \quad L = \frac{V_R}{(\pi d^2 / 4)} = \frac{4}{\pi d^2} y_{A0} F_0 \int_0^{x_A} \frac{1}{(-r_A)} dx \quad (3)$$

$$\text{式中, 组分 A 的初始摩尔分数 } y_{A0} : y_{A0} = \frac{A}{A+1} = \frac{1}{1+0.5} = \frac{2}{3} \quad (4)$$

下面推导压力 p_A 、 p_S 与转化率 x_A 的关系。由化学计量关系得

$$\begin{array}{ccccc} 2A & \longrightarrow & S & & I \\ x_A = 0: & F_{A0} & 0 & F_I & F_{t0} = F_{A0} + F_I \\ x_A: & F_{A0}(1-x_A) & F_{A0}x_A/2 & F_I & F_t = 3n_{A0} + F_I \end{array}$$

$$\text{得 } C_A = \frac{F_{A0}(1-x_A)}{v} \quad (5)$$

$$C_S = \frac{F_{A0}x_A/2}{v} \quad (6)$$

$$\text{其中, 体积流量 } v \text{ 为: } v = v_0(1+\varepsilon x_A) \frac{p}{p_0} \frac{T}{T_0} \quad (7a)$$

$$\text{对于恒温、恒压情况 } (T=T_0, p=p_0), v = v_0(1+\varepsilon x_A) \quad (7b)$$

$$\text{式中, 膨胀因子的计算式为: } \delta = \frac{1-2}{2} = -\frac{1}{2} \quad (8)$$

$$\varepsilon = y_{A0}\delta \quad (9)$$

将式 (7b) 代入 (5)、(6), 得

$$C_A = \frac{F_{A0}(1-x_A)}{v_0(1+\varepsilon x_A)} = C_{A0} \left(\frac{1-x_A}{1+\varepsilon x_A} \right) \quad (10)$$

$$C_S = \frac{F_{A0}x_A/2}{v} = \frac{1}{2} C_{A0} \left(\frac{x_A}{1+\varepsilon x_A} \right) \quad (11)$$

由理想气体状态方程及式 (10) 和 (11), 得

$$p_A = C_A RT = C_{A0} RT \left(\frac{1-x_A}{1+\varepsilon x_A} \right) = p_{A0} \left(\frac{1-x_A}{1+\varepsilon x_A} \right) \quad (12)$$

$$p_S = C_S RT = \frac{1}{2} C_{A0} RT \left(\frac{x_A}{1+\varepsilon x_A} \right) = \frac{1}{2} p_{A0} \left(\frac{x_A}{1+\varepsilon x_A} \right) \quad (13)$$

因此, 将式 (12) 和式 (13) 代入式 (1), 得到 $(-r_A) = f(x_A)$, 再结合式 (3) 可计算已知出口转化率时的管长或已知管长时的出口转化率。

程序说明

(a) **PFRlength.m** (已知 L , 计算 x_A) 将式 (12) 和式 (13) 代入式 (1), 得到 $(-r_A) = f(x_A)$ 后, 再代入式 (3) 进行数值积分求 L 。程序中, **func()** 定义式 (3) 中积分项的被积函数, 函数 **Rate(xA)** 定义速率方程 (1)。式 (3) 中的积分项采用自适应 Lobatto 求积函数 **quadl()**, 积分限为 $[0, x_A]$ 。

平衡转化率 x_{Ae} 的计算: 令式 (1) 的反应速率为 0, 即 $(-r_A) = f(x_{Ae}) = 0$, 即可求得平衡转化率 x_{Ae} 。这是关于 x_{Ae} 的单变量非线性方程, 用函数 **fzero()** 求解, 该函数的用法可参阅 3.3.2。

(b) **PFRconv.m** (已知 x_A , 计算 L) 可直接利用 (a) 的计算结果 $x_A \sim L$, 通过插值求得 $L=5$ 时的 x_A 值。下面介绍另一种方法——同时利用单变量非线性方程求解函数 **fzero()** 和数值积分函数 **quadl()**, 即

$$\text{将式 (3) 变换为 } f(x_A) = L - \frac{4}{\pi d^2} y_{A0} F_0 \int_0^{x_A} \frac{1}{(-r_A)} dx = 0 \quad (14)$$

式(14)是关于 x_A 的单变量非线性方程, 可用 `fzero()` 求解, 其中积分项则用 `quadl()` 计算。程序 `PFRconv.m` 见光盘。

程序清单 *PFRlength.m*

```
function PFRlength
clear all; clc
global PA0 eps
d = 0.1; F0 = 10; P = 1; % d: 反应器内经(m), F0: 进气速率(kmol/h), P: 反应压力(atm)
yA0 = 2/3; PA0 = yA0 * P; % yA0: A 的初始摩尔分数, PA0: A 的初始分压(atm)
delta = -1/2; eps = yA0 * delta; % eps: 膨胀因子
xA = [0.1:0.1:0.5]; % xA: A 的摩尔转化率

% 计算不同转化率下的管长
for i = 1:length(xA)
    I = quadl(@func, 0, xA(i)); % 用自适应 Lobatto 法求积, 积分限为[0, xA]
    L(i) = 4*yA0*F0/(pi*d^2)*I; % L: 管长, m
end
xAe = fzero(@Rate, 0.5); % 平衡转化率

% 结果显示
fprintf('Results:\n\txA:\t'), fprintf('\t%.2f', xA), fprintf('\n\txL (m):')
fprintf('\t%.2f', L), fprintf('\n\txAe = %.3f', xAe)

% -----
function f = func(xA)
f = 1./Rate(xA);

% -----
function r = Rate(xA)
global PA0 eps
PA = PA0 * (1-xA)/(1+eps*xA);
PS = PA0/2 * xA/(1+eps*xA);
r = 116 * (PA.*PA - PS/1.27);
```

计算结果 (a) 由 `PFRlength.m` 算得 $x_A = [0.10 \ 0.20 \ 0.30 \ 0.40 \ 0.50]$, $L(m) = [1.83 \ 4.17 \ 7.46 \ 13.02 \ 42.52]$, $x_{Ae} = 0.503$ 。可见, 当 $x_A > 0.4$ 以后, 所需反应管长剧增, 这是由于在接近平衡转化率 ($x_{Ae} = 0.503$) 处, 反应速率很慢的缘故。因此, 设计反应器时应取 $x_A < 0.4$, 这时, 为充分利用原料 A 可采用循环流程, 即将产物 S 分离后将未反应的原料 A 返回反应器继续反应。(b) 由 `PFRconv.m` 算得: 当 $L = 5m$ 时, $x_A = 0.2291$ 。

若利用 (a) 的结果: $x_A = [0.10 \ 0.20 \ 0.30 \ 0.40 \ 0.50]$, $L = [1.83 \ 4.17 \ 7.46 \ 13.02 \ 42.52]$, 直接用 `spline()` 容易插值得到当 $L = 5m$ 时, $x_A = 0.2291$ 。

【例 3-14】等温管式反应器的空时计算^[6]

某气相反应 $A \rightarrow 3R$ 在等温管式反应器内进行, 反应条件为 $T=215^\circ\text{C}$, $p=506.625\text{kPa}(5\text{ atm})$,

系统保持恒压。已知反应速率为 $(-r_A) = kC_A^{1/2}$ (1)

式中, $k = 0.01$ 。为了控制反应, 向反应物 A 中加入一定惰性气体 I, 两者的摩尔比为 A : I = 1 : 1。已知出口转化率为 0.8, 试计算 PFR 反应器的空时 (space time)。

模型

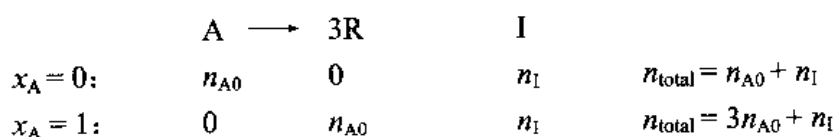
等温管式反应器模型为 $\tau = C_{A0} \int_0^{x_A} \frac{1}{(-r_A)} dx$ (2)

反应过程中有体积变化, 设膨胀因子为 ε_A , 则对于线性膨胀

$$V = V_0(1 + \varepsilon_A x_A) \quad (3)$$

$$C_A = \frac{n_A}{V} = \frac{n_{A0}(1 - x_A)}{V_0(1 + \varepsilon_A x_A)} = \frac{C_{A0}(1 - x_A)}{1 + \varepsilon_A x_A} \quad (4)$$

下面计算膨胀因子:



由理想气体状态方程, 得 $V_{x_A=0} = \frac{(n_{A0} + n_I)RT}{p}$

$$V_{x_A=1} = \frac{(3n_{A0} + n_I)RT}{p}$$

$$\therefore \varepsilon_A = \frac{V_{x_A=1} - V_{x_A=0}}{V_{x_A=0}} = \frac{2n_{A0}}{n_{A0} + n_I}$$

又已知 $n_{A0} / n_I = 1$

$$\therefore \varepsilon_A = 1 \quad (5)$$

$$C_{A0} \text{ 也由理想气体状态方程计算 } C_{A0} = \frac{y_{A0}P}{RT} \quad (6)$$

式中 $y_{A0} = 0.5$ (进料气中 A 的摩尔分数)。

至此, 根据 (1)、(4) ~ (6) 式, 可由 (2) 求得 τ 。

程序说明 采用自适应 Lobatto 求积函数 `quadl()` 进行积分。函数 `funct()` 定义式 (2) 中的被积函数, 其中浓度 C_A 由式 (4) 计算, $(-r_A)$ 由式 (1) 计算。

程序清单 PFRSpaceTime.m

```
function PFRSpaceTime % 等温管式反应器的空时计算
clear all; clc
k = 0.01; T = 488.15; yA0 = 0.5; % T: K, yA0: 进料气中 A 的摩尔分数
P = 5; R = 0.082; CA0 = yA0*P/(R*T); % P: atm, R: 理想气体常数, CA0: mol/liter
epsilon = 1; xAf = 0.8;
a = 0; b = xAf; d = (b-1)/100; % the interval: [a, b], the step: d
I = quadl(@funct, a, b, [], [], CA0, k, epsilon); % 自适应 Lobatto 求积法(高阶)
tau = CA0*I;
fprintf('tSpace time: %.1f (s)', tau)

% -----
```

```
function f = func(xA, CA0, k, epsilon)
CA = CA0*(1-xA)/(1+epsilon*xA);
rate = k*sqrt(CA);
f = 1./rate;
```

计算结果 PFR 的空时 (Space time) τ 为 33.2 s。

【例 3-15】 计算填料吸收塔的总传质单元数 N_{OG} [7]

逆流操作的填料吸收塔, 在温度 20℃、压力 101.325 kPa (1 atm) 的条件下, 用水洗涤含有 5.5% SO_2 的空气, 使 SO_2 下降到 0.5%。当气液比 (L/G) 为 40 时, 求取以气相浓度为基准的总传质单元数 N_{OG} 。若以气相浓度为基准的总传质单元高度 $H_{OG} = 0.69$ m, 进而求填料塔高度。进塔水中不含 SO_2 , 20℃ 时水对 SO_2 的溶解度数据列于表 3-5 中。

表 3-5 20℃ 时水对 SO_2 的溶解度数据

$x \times 10^3$	1.96	1.40	0.846	0.562	0.422	0.281	0.141	0.056
$y \times 10^3$	51.3	34.2	18.6	11.2	7.63	4.21	1.58	0.658

模型 以气相浓度为基准的总传质单元数为
$$N_{OG} = \int_{y_2}^{y_1} \frac{dy}{y - y^*} \quad (1)$$

填料吸收塔操作线方程为
$$G \left(\frac{y}{1-y} - \frac{y_2}{1-y_2} \right) = L \left(\frac{x}{1-x} - \frac{x_2}{1-x_2} \right) \quad (2)$$

气液平衡关系: 由离散平衡数据 (表 3-4) 确定 y^* 与 x 的函数关系。

程序说明 主程序采用 `quadl()` 进行数值积分; 函数 `func()` 定义式 (1) 中的被积函数, 它先由操作方程 (2) 确定 $x = f_1(y)$, 再由三次样条拟合函数 `sp` 确定 y^* 与 x 的函数关系: $y^* = f_2(x)$, 这样就确定了 y^* 与 y 的关系: $y^* = f_3(y)$, 由此, 式 (1) 中的被积函数只与 y 有关, 可进行数值积分。程序中以 `yeq` 表示 y^* 。该程序与例 3-12 程序 `DistStagesCal.m` 相似。

程序清单 见光盘中的 `AbsPackedHeight.m`。

计算结果 总传质单元数为 3.59, 填料塔高度为 2.48 m。

【例 3-16】 反应器停留时间分布的混合特性 [8]

在 $t = 0$ 时刻, 在一容器入口处突然向流进容器的流体脉冲注入一定量的示踪剂, 同时在容器出口处测量流出物料中示踪剂浓度随时间的变化, 实验数据如下表。

t/s	0	20	40	60	80	100	120	140	160	180	200
$C \text{ (kmol/m}^3\text{)} \times 10^3$	0	0	0	0	0.4	5.5	16.2	11.1	1.7	0.1	0

试计算流体在容器中的平均停留时间以及扩散准数。

数学模型

平均停留时间 t_m :
$$t_m = \frac{\int_0^\infty C t dt}{\int_0^\infty C dt} \quad (1)$$

方差 σ^2 :
$$\sigma^2 = \frac{\int_0^\infty C t^2 dt}{\int_0^\infty C dt} - t_m^2 \quad (2)$$

扩散特征数 $D_L/(uL)$ 为:
$$\left(\frac{D_L}{uL}\right) = \frac{\sigma^2}{2t_m^2} \quad (3)$$

对于等时间间隔的实验数据, 式 (1) 和式 (2) 可写成有限差分格式:

$$t_m = \frac{\sum C_i t_i}{\sum C_i} \quad (4)$$

$$\sigma^2 = \frac{\sum C_i t_i^2}{\sum C_i} - t_m^2 \quad (5)$$

程序说明 首先, 用 `spline()` 和 `spaps()` 分别进行插值和拟合, 并画图观察样条插值/拟合效果 (如图 3-6), 发现此处函数 `spaps()` 优于 `spline()`, 因此程序最后按 `spaps()` 的拟合多项式计算式 (1) 和 (2) 中的积分项。接着, 程序直接用式 (1)、(2) 和 (3) 计算, 其中 C 采用 `spaps()` 拟合, 积分式中的无穷大用最后一组数据的对应时刻 ($t=200\text{s}$) 代替。最后, 程序还用式 (4)、(5) 和 (3) 进行计算, 尽管在这里结果还是很满意, 但是, 它要求实验数据必须是等时间间隔, 而且用到差分格式, 当间隔较大时误差也比较大。无疑, 前一种方法 (直接积分法) 比较好。

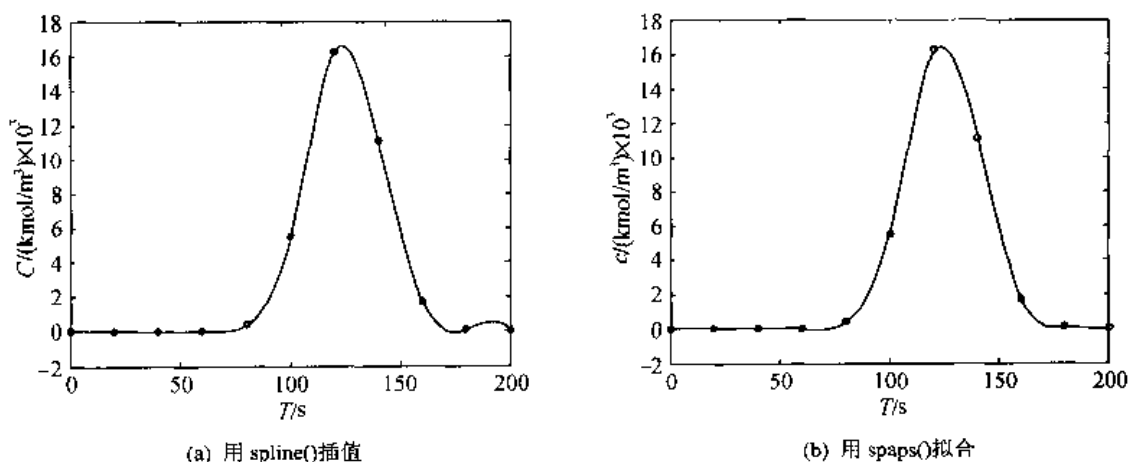


图 3-6 浓度随时间的变化曲线

程序清单 见光盘中的 `RTD1.m`。

计算结果 用式 (1)、(2) 和 (3) 计算得: $t_m = 124.9$, $\sigma^2 = 274.6$, $\left(\frac{D_L}{uL}\right) = 0.0088$;
用式 (4)、(5) 和 (3) 计算得: $t_m = 124.9$, $\sigma^2 = 272.4$, $\left(\frac{D_L}{uL}\right) = 0.0087$ 。

【例 3-17】用微分法进行动力学数据分析^[6]

反应物 A 在一等温间歇反应器中发生的反应为: $A \rightarrow \text{产物}$, 测量得到的反应器中不同时间下反应物 A 的浓度 C_A 如表 3-6 所示。试根据表中数据确定其反应速率方程。

表 3-6 动力学数据

时间 t/s	0	20	40	60	120	180	300
$C_A/(\text{mol/L})$	10	8	6	5	3	2	1

动力学模型

$$-r_A = kC_A^n \quad \text{mol}/(\text{L} \cdot \text{s})$$

即 为便于用函数 `polyfit()` 进行线性拟合, 将上面的动力学方程变形: $\ln(-r_A) = n \ln C_A + \ln k$,
 $y = p(1)x + p(2)$

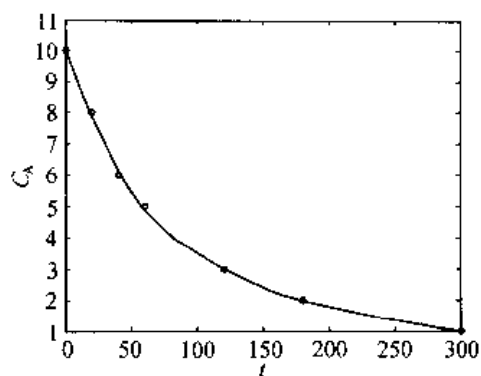


图 3-7 实验数据的最小二乘拟合

式中, $y = \ln(-r_A)$; $x = \ln C_A$; $p(1) = n$;
 $p(2) = \ln k$ 。

程序说明 先用最小二乘样条拟合函数 `spap2()` 拟合得到 B 样条函数 `sp`, 再用 `finder()` 计算 `sp` 的导函数 `pp`, 然后用函数 `fval()` 计算 `pp` 在 t 处的导数值(即反应速率) dC_A/dt 。通过反复调整参数 `knots` 和 `K`, 可使函数 `spap2()` 对数据拟合得更好, 本例在 `knots = 3` 和 `K = 3` 时拟合效果最好, 通过画图可以直观地观察其拟合质量(如图 3-7)。最后, 在拟合满意后用 `polyfit()` 进行单变量的线性拟合以计算动力学参数 k 和 n 。

程序清单 *KineticDataFit.m*

```
function KineticDataFit
clear all; clc

% 动力学数据
t = [0 20 40 60 120 180 300]; % t -- vector of reaction time
CA = [10 8 6 5 3 2 1]; % CA -- concentration vector for reactant A

% 用最小二乘样条拟合法计算微分 dCA/dt--使用不经过实验点的 B 样条插值函数
knots = 3; K = 3; % 三次 B 样条
sp = spap2(knots, K, t, CA)
sp = spap2(newknt(sp), K, t, CA);
pp = finder(sp)
dCAdt = fnval(pp, t) % 计算 B 样条函数的导函数 pp 及其在 t 处的导数值

% 绘制图形
ti = linspace(t(1), t(end), 200); CAi = fnval(sp, ti)
plot(t, CA, 'ro', ti, CAi, 'b-'), xlabel('t'), ylabel('C_A')
figure, fnplt(pp), xlabel('t'), ylabel('dC/dt')

% 线性拟合
rA = dCAdt; y = log(-rA); x = log(CA);
p = polyfit(x, y, 1);
k = exp(p(2)), n = p(1)
```

计算结果 $k = 0.0053, n = 1.39$

本例主要介绍数值微分在动力学数据分析中的应用, 即用微分法由动态浓度数据求取反应速率, 进而求取动力学参数。为简单起见, 这里用线性拟合方法求取动力学参数, 但需要注意的是, 这种方法由于将原来的非线性模型进行线性化, 最终得到的结果从统计方面来说, 是不严格的, 只能算是近似的。因此, 应尽量使用最优化方法(如非线性最小二乘法)直接

由非线性模型估计其参数。本例的最优化参数估计方法请参阅第7章[例7-1]。此外，数值微分从本质上精度要比数值积分差，所以动力学数据应尽量使用积分法分析。

3.3 代数方程（组）的数值解法

3.3.1 线性代数方程组的解法

已知线性代数方程组： $AX = b$ ，则解为 $X = A^{-1}b$ ，用 MATLAB 语言表示为： $X = \text{inv}(A)*b$ ，其中 MATLAB 的矩阵求逆函数为 $\text{inv}()$ 。也可以用左除符号“ \backslash ”求解： $X = A \backslash b$ 。

例 求解下列线性方程组：

$$\begin{cases} 4x_1 - x_2 + x_3 = 5 \\ -18x_1 + 3x_2 - x_3 = -15 \\ x_1 + x_2 + x_3 = 6 \end{cases}$$

用 MATLAB 计算如下：

```
>>a=[4 -1 1; -18 3 -1; 1 1 1];
```

```
>>b=[5 -15 6]';
```

```
>>x = inv(a)*b
```

或

```
>>x = a\b
```

3.3.2 非线性代数方程（组）的解法

单变量非线性方程： $f(x) = 0$ (3-18)

(1) 用函数 $\text{fzero}()$ 求解

单变量非线性方程的数值解法主要有二分法 (bisection)、不动点迭代法、Newton 迭代法、割线法 (secant) 和 Muller 迭代法等。MATLAB 用函数 $\text{fzero}()$ 求解单个代数方程，该函数结合使用二分法、割线法 (secant) 和可逆二次内插法 (inverse quadratic interpolation)。

函数 $\text{fzero}()$ 的用法：

```
x = fzero(@fun, x0)
```

```
x = fzero(@fun, x0, options, p1, p2, ...)
```

其中， fun 为自定义函数，即定义 $f(x) = 0$ 中的 $f(x)$ ； x_0 为迭代初值； x 为方程的根。

(2) 对于下列多项式方程，用函数 $\text{roots}()$ 求解更有效。

$$f(x) = C(1) \cdot x^n + C(2) \cdot x^{n-1} + \dots + C(n) \cdot x + C(n+1) = 0 \quad (3-19)$$

函数 $\text{roots}()$ 的用法：

```
x = roots(C) % 用于求解多项式 C(1)*X^N + ... + C(N)*X + C(N+1) = 0 的根
```

其中， C 为多项式的系数向量， x 为方程的根。

多变量非线性系统（方程组）： $f(x) = 0$ (3-20)

MATLAB 用函数 $\text{fsolve}()$ 求解非线性代数方程组，其算法是最小二乘法。

函数 $\text{fsolve}()$ 的用法：

```
x = fsolve(@fun, x0)
```

```
x = fsolve(@fun, x0, options, p1, p2, ...)
```

其中， fun 为自定义函数，即定义非线性方程组 $f(x) = 0$ 中的 $f(x)$ ，该函数应该返回一个列向量； x_0 为迭代初值向量； x 为方程的根向量。

注 若某个问题有多个解，则实际获得的解将取决于给定的初值。许多化工过程有“多解”，即有一些稳态点。实际上，稳态的获得将取决于动态考虑，例如开车的过程。对单变量系统，建

议画出 $f(x)$ ，以观察计算结果是否合理。

【例 3-18】 计算方程 $x^2 - 2x - 3 = 0$ 的根。

程序说明 函数 `func()` 定义 $f(x) = x^2 - 2x - 3$ ，主程序分别用 `fzero()` 和 `roots()` 求解 $f(x) = 0$ 的根。

程序清单 *xFzero_Roots.m*

```
function xFzero_Roots    % to demonstrate how to use function fzero() and root()
clear all;  clc

% By using fzero() with different initial values of independent variable
x0 = 0;  x1 = fzero(@func, x0)
x0 = 2;  x2 = fzero(@func, x0)

% By using roots()
c = [1    2   -3];
x3 = roots(c)

% -----
function f = func(x)
f = x^2 - 2*x - 3;
```

计算结果 用 `fzero()` 计算时，当初值分别取 $x_0 = 0$ 和 $x_0 = 2$ 时，结果分别为 $x_1 = -1$ 和 $x_2 = 3$ ，这表明用 `fzero()` 获得的解取决于初值，初值不同，得到的根不同，一般得到靠近给定初值的根。由于这是多项式方程，用 `roots()` 可求到全部的根 $x_3 = [-1 \ 3]$ ，而用 `fzero()` 则很难得到全部的根。因此，对于多项式方程，应该选用 `roots()` 求解，而不用 `fzero()`。

【例 3-19】 求解非线性方程组：
$$\begin{cases} x_1 - 4x_1^2 - x_1x_2 = 0 \\ 2x_2 - x_2^2 + 3x_1x_2 = 0 \end{cases}$$

程序说明 函数 `NonlinEqs()` 定义所要求解的非线性方程组，供函数 `fsolve()` 调用。

程序清单 *xFsolve.m*

```
function xFsolve
clear all;  clc

x0 = [1 1]';  x1 = fsolve(@NonlinEqs, x0)
x0 = [-0.1 2]';  x2 = fsolve(@NonlinEqs, x0)

% -----
function f = NonlinEqs(x)
f(1) = x(1) - 4*x(1)*x(1) - x(1)*x(2);
f(2) = 2*x(2) - x(2)*x(2) + 3*x(1)*x(2);
```

计算结果 给定初值 $x_0 = [1 \ 1]'$ 时，解为 $x_1 = [0.2500 \ 0.0000]'$ ； $x_0 = [-0.1 \ 2]'$ 时，解为 $x_2 = [-0.1429 \ 1.5714]'$ 。可见，用 `fsolve()` 求解非线性方程组时，给定的初值不同，结果也就不同。

3.3.3 化工应用实例

【例 3-20】 由状态方程 p - V - T 关系计算^[7] 在 945.36 kPa(9.33atm)、300.2 K 时，容器中

充以 2 mol 氮气, 试求该容器的体积。已知氮气的范德华常数 $a = 4.17 \text{ atm}\cdot\text{L}^2/\text{mol}^2$, $b = 0.0371 \text{ L/mol}$ 。

数学模型 范德华方程为: $\left(p + \frac{an^2}{V^2}\right)(V - nb) = nRT$

程序说明 将范德华方程变为 $f(V) = \left(p + \frac{an^2}{V^2}\right)(V - nb) - nRT = 0$, 这是关于 V 的单变量非线性方程 (p, T 已知), 该方程在程序中由函数 `pVTeq()` 定义, 其根 V 由函数 `fzero()` 求解。初值 V_0 根据理想气体状态方程确定。

程序清单 见光盘中的 `pVT.m`。

计算结果 $V = 5.0028$ 。

【例 3-21】简单蒸馏计算—简单蒸馏时, 某时刻釜残液量与低沸点组成 x 之间的关系式为

$$\ln \frac{F_0}{F} = \frac{1}{\alpha - 1} \left(\ln \frac{x_0}{x} + \alpha \ln \frac{1-x}{1-x_0} \right)$$

对于苯-甲苯物系, 相对挥发度 $\alpha = 2.5$, 开始时物系中含苯 60%, 甲苯 40%。试求当蒸馏至残液量为原加料量的一半时残液中的苯含量。

程序说明 将方程变为 $f(x) = \ln \frac{F_0}{F} - \frac{1}{\alpha - 1} \left(\ln \frac{x_0}{x} + \alpha \ln \frac{1-x}{1-x_0} \right) = 0$

这是关于 x 的单变量非线性方程 (其他参数已知), 该方程由函数 `DistEq()` 定义, 并以物系苯含量作为迭代初值 x_0 , 即 $x_0 = 0.6$, 用 `fzero()` 求解方程的根 x 。

程序为 `BatchDist.m` (见配套光盘), 计算结果为: $x = 0.4565$ 。

相平衡计算

这里主要举例汽液平衡计算, 包括泡点计算、露点计算和闪蒸计算等三方面。

泡点计算主要求取液相混合物在一定压力下的沸点即泡点, 或一定温度下的蒸汽压, 以及平衡汽相组成。输入的独立变量为 x 以及 T 或 p 中的任一个, 要求输出另一个以及 y 。

露点计算主要求取汽相混合物在一定压力下的露点, 或一定温度即露点下的压力, 以及平衡液相组成。输入的独立变量为 y 以及 T 或 p 中的任一个, 要求输出另一个以及 x 。

闪蒸计算主要求取一定温度压力下, 混合物分相后的汽液两相组成。输入的独立变量为 T, p 和混合物的总组成 z (进料组成), 要求输出 x 和 y 。

这些问题都是单变量非线性方程的求解问题。

【例 3-22】计算混合液体的泡点及平衡蒸汽组成^[7]

计算在 101.325 kPa (1 大气压) 下 0.5 (摩尔分数) 苯、0.3 (摩尔分数) 甲苯、0.2 (摩尔分数) 乙苯混合物的泡点及平衡蒸汽组成。已知每一纯组分 i 的饱和蒸汽压 p_i^0 与绝对温度 T 有下列关系:

$$\lg_{10}(p_i^0) = a_i - b_i/T \quad (1)$$

式中, p_i^0 的单位是 mmHg; T 的单位为 K。系数 a_i 和 b_i 如下表。

i	组分	a_i	b_i
1	苯	7.84135	1750
2	甲苯	8.08840	1985
3	乙苯	8.11404	2129

模型

根据 Laoult (拉乌尔) 定律, 各组分蒸汽分压 p_i 为 $p_i = x_i p_i^0$ (2)

总压为 $p = \sum_{i=1}^n p_i$, 即 $\sum_{i=1}^n y_i - 1 = 0$ (3)

式中 $y_i = \frac{p_i}{p}$ (4)

将式 (2) 代入式 (4) 得 $y_i = \frac{x_i p_i^0}{p}$ (5)

由式 (1)、式 (3) 和式 (5) 可知, 当 x_i 和 p 已知时式 (3) 是关于 T 的单变量非线性方程。

程序说明 函数 Equation() 定义方程 (3), 而函数 P0() 定义方程 (1), 用于计算各组分饱和蒸汽压。主程序用 fzero() 求解单变量非线性方程 (3), 并以纯苯的沸点作为其计算的初值。因为混合液中苯含量最多, 故以它的沸点为计算初值比较合适。

程序清单 BoilingPoint.m

```
function BoilingPoint
clear all; clc
x = [0.5 0.3 0.2]; a = [7.84135 8.08840 8.11404]; % x: 液相组成
b = [1750 1985 2129]; P = 760; T0 = 353; % P: 大气压(mmHg), T0: 温度(K)
T = fzero(@Equation, T0, [], x, a, b, P);
y = x.*P0(T, a, b)./P;
fprintf('\n\n Results: '), fprintf('\n\t 混合物沸点为: %.2f %s\n', T, 'K')
fprintf('\t 平衡蒸汽组成为: '), disp(y)

% -----
function f = Equation(T, x, a, b, P)
y = x.*P0(T, a, b)./P;
f = sum(y) - 1;

% -----
function f = P0(T, a, b)
f = exp(log(10)*(a-b./T));
```

计算结果 混合物沸点为 $T = 366.3267$, 气相组成为 $y = [0.7627 \ 0.1845 \ 0.0528]$ 。

【例 3-23】等温闪蒸计算^[9]

如图 3-8 所示, 进料组成如表 3-7 所示的四组分烷烃混合物在 1380 kPa 下加入闪蒸器 ($p_L = p_V$) 进行等温闪蒸 ($T_L = T_V$), 若气化率为 40%, 试问混合物应预热到多高的温度? 假设仅形成一个液相, 且具有理想溶液的性质, 即气液平衡常数 K 只与温度 T 和压力 p 有关, 其函数关系为:

$$\ln(Kp) = a + \frac{b}{T} + cT \quad (1)$$

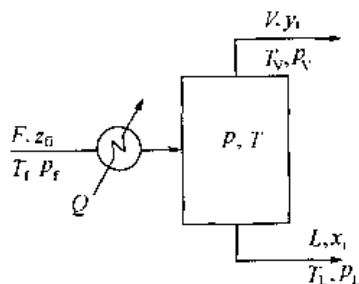


图 3-8 等温闪蒸

式中, 系数 a, b 和 c 见表 3-7 所示。

表 3-7 进料组成

序号	组分	z_i	a	b	c
1	C ₂	0.08	15.57	-1793.4	-4.94×10 ⁻³
2	C ₃	0.22	17.14	-2843.0	-6.27×10 ⁻³
3	n-C ₄	0.53	17.83	-3007.2	-6.67×10 ⁻³
4	n-C ₅	0.17	17.91	-3412.9	-6.12×10 ⁻³

模型

总质量衡算 $F = L + V$ (2)

各组分质量衡算 $Fz_i = Lx_i + Vy_i$ (3)

摩尔分数总和方程 $\sum_{i=1}^{N_C} x_i = 1$ (4)

$$\sum_{i=1}^{N_C} y_i = 1 \quad (5)$$

而多组分汽-液平衡关系为 $y_i = K_i x_i$ (6)

将式 (2) 和式 (6) 代入式 (3), 整理得 $x_i = \frac{z_i}{1 + \beta(K_i - 1)}$ (7)

或 $y_i = \frac{K_i z_i}{1 + \beta(K_i - 1)}$ (8)

式中, β 为气化率, 即 $\beta = \frac{V}{F}$ 。

这里给定 p 和 β , 故通过式 (1)、式 (4)、式 (5)、式 (7)、式 (8) 可求得 T 。但若直接应用式 (4) 和式 (5), 则收敛很慢。而把式 (4) 和式 (5) 结合成如下一个函数, 会得到较好的收敛特性。

$$f(T) = \sum_{i=1}^{N_C} y_i - \sum_{i=1}^{N_C} x_i = 0 \quad (9)$$

将式 (7) 和式 (8) 代入式 (9) 得 $f(T) = \sum_{i=1}^{N_C} \frac{(K_i - 1)z_i}{1 + \beta(K_i - 1)} = 0$ (10)

这样, 根据式 (1) 和式 (10) 可迭代求得 T 。

程序说明 由式 (1) 和式 (10) 可知, 式 (10) 是关于 T 的单变量非线性方程 (p 和 β 已知), 因此, 程序也使用 `fzero()` 求解。程序中, 函数 `func()` 定义式 (10), 函数 `K()` 定义式 (1)。

注 本程序需要给定合适的温度初值 (起始估算值)。在某些情况下, 选择不适宜的初值可能使混合物处于过热或过冷状态, 因此不发生闪蒸。所以, 初值温度应该避免出现过热或过冷状态, 详见文献[9]。这里, 初值取 $T_0 = 300^\circ\text{C}$ 可得到满意的结果。

程序清单 *Flash.m*

```
function Flash
clear; clc
global a b c P
P = 1380; beta = 0.40; % P: kPa, beta: 气化率
```

```

z = [0.08 0.22 0.53 0.17]; % 进料组成
a = [15.57 17.14 17.83 17.91]; b = [-1793.4 -2843.0 -3007.2 -3412.9];
c = [-4.94 -6.27 -6.67 -6.12]*1e 3;
T0 = 355.6;
T = fzero(@func, T0, [], z, beta)
x = z./(1+beta*(K(T)-1)), y = K(T).*x

% -----
function f = func(T,z,beta)
numer = (K(T) - 1).*z; denom = 1 + beta*(K(T)-1);
f = sum(numer./denom);

% -----
function f = K(T)
global a b c P
f = exp(a+b./T+c.*T) ./P;

```

计算结果 $T=370.9009$, $x=[0.0293, 0.2270, 0.5261, 0.2176]$, $y=[0.1561, 0.2095, 0.5359, 0.0986]$ 。

符号说明

a, b, c	K 表达式中的系数	T	闪蒸温度, K
F	进料量, mol	V	闪蒸后汽相量, mol
K	汽液平衡常数	x_i	闪蒸后组分 i 的液相组成 (摩尔分数)
L	闪蒸后液相量, mol	y_i	闪蒸后组分 i 的气相组成 (摩尔分数)
N_C	系统中的组分数	z_{fi}	组分 i 的进料组成 (摩尔分数)
p	闪蒸压力, kPa	β	气化率

下标

i 任意组分 i

【例 3-24】 求解绝热连续搅拌槽反应器的转化率^[10]

如图 3-9 所示, 预热到 T_0 的含有反应物 A 的溶液原料, 以一定的流量 Q , 加入到容积为 V_R 的搅拌槽反应器中进行绝热反应。反应混合物连续排出。A 的进、出口浓度分别为 C_{A0} 和 C_A 。反应溶液的密度为 ρ , 比热容为 c_p 。槽内及出口温度为 T , 反应速度为

$$-r_A = kC_A^2 \quad (1)$$

$$\text{式中} \quad k = k_0 \exp\left(-\frac{E}{RT}\right) \quad (2)$$

已知数据: $T_0 = 450$ K, $C_{A0}(-H_r)/(\rho c_p) = 250$ K, $E/R = 10000$ K, $k_0 C_{A0} = e^{20}$, $\tau = V_R/Q = 0.25$ h。试求反应转化率。

模型

对组分 A 进行物料衡算 (稳态):

$$QC_{A0} - QC_A - (-r_A)V_R = 0 \quad (3)$$

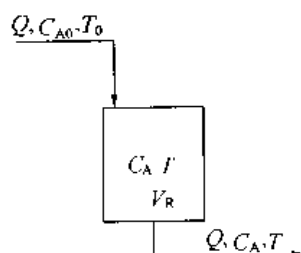


图 3-9 绝热式 CSTR

将 $C_A = C_{A0}(1-x)$ 及式(1)和(2)代入(3), 化简得

$$k_0 C_{A0} (1-x)^2 \tau \exp\left(-\frac{E}{RT}\right) - x = 0$$

即
$$0.25(1-x)^2 \exp\left(20 - \frac{10000}{T}\right) - x = 0 \quad (4)$$

对组分 A 进行热量衡算 (绝热): $Q \rho c_p (T_0 - T) + (-\Delta H_r)(-r_A)V_R = 0 \quad (5)$

由(3)得: $(-r_A)V_R = QC_{A0} - QC_A = QC_{A0}x$

代入(5)并整理得
$$T - T_0 = \frac{(-\Delta H)C_{A0}x}{\rho c_p}$$

即
$$T = T_0 + 250x \quad (6)$$

方程(4)和(6)就是模型方程。

程序说明 由(6)代入(4)即变成关于 x 的单变量非线性方程, 因此, 也用函数 `fzero()` 求解。程序中, 函数 `NonlinEq()` 用于定义此单变量非线性方程。

程序清单 见光盘中的 `ConvCSTR.m`。

计算结果 所求的转化率为 $x = 0.8363$ 。

【例 3-25】一水平管道内幂定律流体的流动—估算非牛顿型流体流动所需的管道直径^[9]

一管道内幂定律流体的水平流动: $\rho = 961 \text{ kg} \cdot \text{m}^{-3}$, 质量流量 $G = 6.67 \text{ kg} \cdot \text{s}^{-1}$, 管长 $L = 10 \text{ m}$, $\varepsilon = 5 \times 10^{-6} \text{ m}$, $\Delta p = 15 \text{ kPa}$, $k = 1.8$, $n = 0.64$, $K = 1.48 \text{ N} \cdot \text{s}^{2-n} \cdot \text{m}^{-2}$ 。试用普遍化的雷诺数法估算此非牛顿流动情况所需的管道直径。

数学模型

普遍化的雷诺数法定义式为:
$$Re_{\text{gen}} = \frac{D^{n'} u^{2-n'} \rho}{8^{n'-1} K'} \quad (1)$$

式中, n' 和 K' 是剪切速率的函数。

对于幂定律流体
$$n' = n \quad (2)$$

$$K' = K \left[\frac{3n+1}{4n} \right]^n \quad (3)$$

普遍化的 Re - f 关系式

层流时 ($Re_{\text{gen}} \leq 2100$)
$$f = \frac{16}{Re_{\text{gen}}} \quad (4)$$

湍流时 ($Re_{\text{gen}} > 2100$)
$$\frac{1}{\sqrt{f}} = \frac{4.0}{n^{0.75}} \lg(Re_{\text{gen}} f^{1/n+2}) - \frac{0.4}{n^{1.2}} \quad (5)$$

管道直径 D 的计算是迭代过程, 其起始估计值按下式估算:

$$D^{(1)} = \left[\frac{32K'L8^{n'-1} \left(\frac{4G}{\pi\rho} \right)^{n'}}{\Delta p} \right]^{1/(3n'+1)} \quad (\text{层流, } k=0) \quad (6)$$

D 的改进估计值
$$D^{(r+1)} = \left[\frac{2f(L+Le) \left(\frac{4G}{\pi} \right)^2}{\rho\Delta p} \right]^{0.2} \quad (7)$$

式中 $(r+1)$ 为迭代次数, 接口的当量长度计算公式为 $Le = \frac{kD}{4f}$ 。 (8)

程序说明 管径的迭代计算过程如下。

- (a) 由式 (2)、式 (3) 计算 n' 和 K' ;
 (b) 假设层流、 $k=0$, 则由式 (6) 可估算得到 D 的起始估计值;
 (c) 由式 (1) 计算 Re_{gen} ;
 (d) 若 $Re_{gen} \leq 2100$, 则由式 (4) 求 f ;
 (e) 若 $Re_{gen} > 2100$, 则由式 (5) 求 f , 由于式 (5) 是单变量非线性方程, 改写为

$$F(f) = \frac{1}{\sqrt{f}} - \frac{4.0}{n^{0.75}} \lg(Re_{gen} f^{1-n/2}) - \frac{0.4}{n^{1.2}} = 0 \quad (9)$$

后, 即可用函数 `fzero()` 求解。

- (f) 由式 (8) 计算 Le ;
 (g) 由式 (7) 计算 D 的改进估计值。

(h) 按收敛判据
$$\left| \frac{D^{(r+1)} - D^{(r)}}{D^{(r+1)}} \right| < \varepsilon \quad (10)$$

判定是否收敛。其中, ε 为收敛精度。

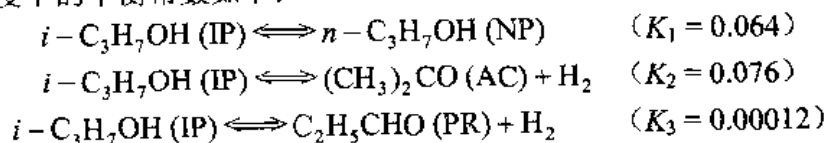
若不满足式 (10), 则重复 (c) ~ (h) 过程, 直至满足为止。

程序清单 见光盘中的 *PipeDiameter.m*。

计算结果 管道直径为 $D = 0.0888$ m, 摩擦因子为 $f = 0.0512$ m。

【例 3-26】化学平衡计算^[1]

异丙醇于 400°C 、 0.1033MPa (1atm) 下在银催化剂上进行脱氢反应生成丙酮, 主副反应及其在反应温度下的平衡常数如下:



后续的分选过程要求反应产物中丙醛的含量不大于 $0.05\text{ mol}\%$ 。

问在所述反应条件下, 产物组成能否满足此要求。

数学模型 由上述三条反应式容易看出, 它们互为独立反应。以 1 mol 异丙醇为基准, 设达到平衡时三个反应的反应量为 x_{e1} 、 x_{e2} 和 x_{e3} , 则由化学计量关系可得到平衡时各组分的物质的量 (摩尔): 异丙醇 $N_{\text{IP}} = 1 - x_{e1} - x_{e2} - x_{e3}$, 正丙醇 $N_{\text{NP}} = x_{e1}$, 丙酮 $N_{\text{AC}} = x_{e2}$, 丙醛 $N_{\text{PR}} = x_{e3}$, 氢 $N_{\text{H}} = x_{e2} + x_{e3}$, 总物质的量 (摩尔) $N_t = 1 + x_{e2} + x_{e3}$ 。于是, 可写出化学平衡方程:

$$\frac{N_{\text{NP}}}{N_{\text{IP}}} = \frac{x_{e1}}{1 - x_{e1} - x_{e2} - x_{e3}} = K_1 \quad (1)$$

$$\frac{N_{\text{AC}} \cdot N_{\text{H}}}{N_{\text{IP}} \cdot N_t} = \frac{x_{e2}(x_{e2} + x_{e3})}{(1 - x_{e1} - x_{e2} - x_{e3})(1 + x_{e2} + x_{e3})} = K_2 \quad (2)$$

$$\frac{N_{\text{PR}} \cdot N_{\text{H}}}{N_{\text{IP}} \cdot N_t} = \frac{x_{e3}(x_{e2} + x_{e3})}{(1 - x_{e1} - x_{e2} - x_{e3})(1 + x_{e2} + x_{e3})} = K_3 \quad (3)$$

程序说明 将式 (1) ~ (3) 变形后得:

$$f_1 = \frac{x_{e1}}{1 - x_{e1} - x_{e2} - x_{e3}} - K_1 = 0 \quad (4)$$

$$f_2 = \frac{x_{e2}(x_{e2} + x_{e3})}{(1 - x_{e1} - x_{e2} - x_{e3})(1 + x_{e2} + x_{e3})} - K_2 = 0 \quad (5)$$

$$f_3 = \frac{x_{e3}(x_{e2} + x_{e3})}{(1 - x_{e1} - x_{e2} - x_{e3})(1 + x_{e2} + x_{e3})} - K_3 = 0 \quad (6)$$

得到的方程 (4) ~ (6) 是关于 x_{e1} 、 x_{e2} 和 x_{e3} 的非线性方程组，可用函数 `fsolve()` 求解。求出 x_{e1} 、 x_{e2} 和 x_{e3} 以后，化学平衡时丙醛的摩尔分数为 $\frac{x_{e3}}{1 + x_{e2} + x_{e3}}$ 。

需注意的是，初值的选择必须满足约束条件： $x_{e1} + x_{e2} + x_{e3} < 1$ ，如 $x_e = [0.3 \ 0.3 \ 0]$ 。

程序清单 ChemEquil.m

```
function ChemEquil          % 求解化学平衡问题
clear all; clc
x0 = [0.3  0.3  0];
x = fsolve(@NonlinEqs, x0)
fprintf('t 化学平衡时丙醛的摩尔分数为: %.3f %s', x(3)/(1+x(2)+x(3))*100, '%')

% -----
function f = NonlinEqs(x)
tmp1 = 1-x(1)-x(2)-x(3); tmp2 = 1 + x(2) + x(3);
f(1) = x(1)/tmp1-0.064;
f(2) = x(2)*(x(2)+x(3))/(tmp1*tmp2)-0.076;
f(3) = x(3)*(x(2)+x(3))/(tmp1*tmp2)-0.00012;
```

计算结果 化学平衡时三反应的反应量分别为 0.0446、0.2580 和 0.0004，丙醛的摩尔分数为 0.032 %。可见，只要反应时间足够长，在题述的反应条件下，丙醛含量不会超过 0.05%。

【例 3-27】非等温 CSTR——求有夹套换热 CSTR 的多稳态解^[12]。

如图 3-10 所示，在一个非等温 CSTR 反应器中进行反应： $A \xrightarrow{k} B$ ，假设流量和持液量（液体容积 V ）不变，夹套的冷却水流量 F_J 和冷却夹套中持液量恒定、混合完全。系统原先处于稳态过程 ($t=0$ 时刻之前)，试确定多稳态解。

已知参数值如下。

$$\begin{aligned} C_{A0} &= 0.55 \text{ lbmol}/\text{ft}^3 \\ F_0 = F &= 40 \text{ ft}^3/\text{h} \\ F_J &= 49.9 \text{ ft}^3/\text{h} \\ V &= 48 \text{ ft}^3 \\ A_H &= 250 \text{ ft}^2 \\ c_p &= 0.75 \text{ Btu}/(\text{lb}_m \cdot ^\circ\text{R}) \\ U &= 150 \text{ Btu}/(\text{h ft}^2 \cdot ^\circ\text{R}) \\ k_0 &= 7.08 \times 10^{10} \text{ h}^{-1} \\ \rho &= 50 \text{ lb}_m/\text{ft}^3 \end{aligned}$$

$$\begin{aligned} C_A &= 0.245 \text{ lbmol}/\text{ft}^3 \\ T_0 &= 530 ^\circ\text{R} \\ T_{J0} &= 530 ^\circ\text{R} \\ V_J &= 12 \text{ ft}^3 \\ c_{pJ} &= 1.0 \text{ Btu}/(\text{lb}_m \cdot ^\circ\text{R}) \\ E &= 30000 \text{ Btu}/(\text{lbmol}) \\ R &= 1.9872 \text{ Btu}/(\text{lbmol} \cdot ^\circ\text{R}) \\ \Delta H &= -30000 \text{ Btu}/\text{lbmol} \\ \rho_J &= 62.3 \text{ lb}_m/\text{ft}^3 \end{aligned}$$

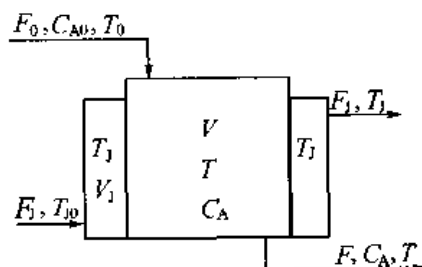


图 3-10 非等温 CSTR

● 1 lbmol (磅摩尔) = 453.6 mol. ● 1 ft = 0.3048 m. ● $T(^{\circ}\text{R}) = 1.8 T(^{\circ}\text{K})$. ● 1 Btu = $1.055 \times 10^3 \text{ J}$. ● 1 lb_m = 0.4536 kg.

模型

$$\text{稳态时组分 A 的物料平衡方程} \quad F_0 C_{A0} - F C_A - V k C_A = 0 \quad (1)$$

$$\text{稳态时反应器中物料的热量平衡} \quad F_0 T_0 - F T - \frac{(-\Delta H) V k C_A}{\rho c_p} - \frac{Q}{\rho c_p} = 0 \quad (2)$$

$$\text{式中} \quad Q = \frac{U A_H}{\rho c_p} (T - T_J) \quad (3)$$

$$\text{反应速率常数} \quad k = k_0 \exp(-E / RT) \quad (4)$$

$$\text{稳态时夹套中冷却流体的热量平衡} \quad F_J (T_{J0} - T_J) + \frac{Q}{\rho_J c_{pJ}} = 0 \quad (5)$$

由式(1)、式(2)和式(5)构成的非线性代数方程组，可用函数 `fsolve()` 求解。

程序说明 `Equations()` 定义由式(1)、(2)和(5)构成的非线性代数方程组，并用 `fsolve()` 求解。该方程组有多个解，程序在温度范围 530~730 °R 内搜索全部稳态解，即用循环语句改变 T_0 的初值（以 10 °R 为增量从 530 °R 不断增加至 730 °R），而 C_{A0} 和 T_{J0} 的初值固定不变（分别是 $C_{A0} = 0.55$ 和 $T_{J0} = 530$ ），以这些初值调用 `fsolve()` 求解（每执行一次循环，求解一次）。把所有搜索得到的解存入单元数组 $y\{i\}$ 中，其部分解如下表所示。

i	y{i}		
	C_A	T	T_J
1	0.5214	537.8548	537.2534
2	0.5214	537.8548	537.2534
3	0.5214	537.8548	537.2534
4	0.5214	537.8548	537.2534
5	0.3302	590.3498	585.7298
...
16	0.0354	671.2783	660.4629
17	0.3302	590.3498	585.7298
18	0.0354	671.2783	660.4629
19	0.3302	590.3498	585.7298
20	0.0354	671.2783	660.4629
21	0.0354	671.2783	660.4629

如果全部列出解，可挑出不同的稳态解（这里有 3 个），但这种方法不理想，若循环次数很多（增量更小些，或温度范围更宽些），需要用程序自动完成，实现过程是：将单元数组 $y\{i\}$ 的每个单元加和，即 $sy(i) = C_A(i) + T(i) + T_J(i)$ ，再将数组 $sy(i)$ 从小到大排列，返回排列后的数组 sy 及原数组各元素下标向量 $i1$ ，排列后再用 `diff()` 计算相邻元素的差值，得到差值向量 dsy ，然后用 `find()` 找出 dsy 中差值大于 $1e-6$ 的对应元素下标 $i2$ ，则排序后的 sy 向量中，非重复数据的下标位置为 $[1, i2]$ ，所以不同的稳态解即为 $y\{i1(i2)\}$ 。

程序清单 *MultipleStateCSTR.m*

```
function MultipleStateCSTR          % 多稳态 CSTR 反应器
clear all; clc
global F0 F CA0 Kc T0 k0 dH rho rhoJ Cp VJ CPJ U Ah FJ TJ0 V

% parameter values
```

```

TJ0 = 530; T0 = 530; F0 = 40; F = 40; FJ = 49.9; CA0 = 0.55; V = 48; VJ = 12;
E = 30000; U = 150; Cp = 0.75; rho = 50; k0 = 7.08e10; R = 1.9872; Ah = 250;
dH = -30000; CPJ = 1.0; rhoJ = 62.3; % dH: 反应热
Ti = T0:10:T0 + 200; % initial conditions

n = length(Ti);
for i = 1:n
    sol = fsolve(@Equations, [CA0,Ti(i),TJ0]); y{i} = sol;
end

% 去掉重复解即得到多稳态解
for i=1:n, sy(i) = sum(y{i}); end
[sy, i1] = sort(sy); dsy = diff(sy); i2 = find(abs(dsy)>1e-6); i2 = [1, i2+1];
fprintf('\n 有%d%s', length(i1(i2)), '个稳态解(CA,T,TJ0): ', y{i1(i2)}) % y: 多稳态解:

% -----
function f = Equations(y)
global F0 F CA0 Kc T0 k0 dH rho rhoJ Cp VJ CPJ U Ah FJ TJ0 V
CA = y(1); T = y(2); TJ = y(3); k = k0*exp(-30000/(1.9872*T)); Q = U*Ah*(T - TJ);
f1 = F0*CA0 - F*CA - V*k*CA;
f2 = F0*T0 - F*T - (dH*V*k*CA + Q)/(rho*Cp);
f3 = FJ*(TJ0 - TJ) + Q/(rhoJ*CPJ);
f = [f1; f2; f3];

```

计算结果 有 3 个稳态解（三个状态）。

C_A	T	T_{J0}
0.5214	537.85	537.25
0.3302	590.35	585.73
0.0354	671.28	660.46

3.4 常微分方程（组）的数值解法

大多数化工过程的常微分方程（ODE）或常微分方程组（ODEs）是非线性的，通常很难得到分析解。因而常微分方程（组）的数值解法在科学研究和工程实际中显得非常重要。

常微分方程（组）包括初值问题（IVP）和边值问题（BVP）两种。若给出未知函数的初始条件，相应的定解问题称为初值问题。若给出未知函数的首末两个端点的条件，这类定解条件称为边值条件，相应的定解问题称为边值问题。

一阶常微分方程初值问题（ODE-IVP）的一般形式：

$$\begin{cases} \frac{dy}{dx} = f(x, y), & a \leq x \leq b \\ y(a) = y_0 \end{cases} \quad (3-21)$$

式中， f 是已知的函数； y_0 是已知的数。

一阶常微分方程组初值问题（ODE-IVPs）的一般形式：

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2, \dots, y_m) \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2, \dots, y_m) \\ \dots\dots \\ \frac{dy_m}{dx} = f_m(x, y_1, y_2, \dots, y_m) \end{cases} \quad (3-22a)$$

I.C. (初始条件)

$$\begin{cases} y_1(a) = y_{10} \\ y_2(a) = y_{20} \\ \dots\dots \\ y_m(a) = y_{m0} \end{cases} \quad (3-22b)$$

上述 ODE-IVPs 问题 (3-22a)、(3-22b) 可写成如下的向量形式

$$\begin{cases} \mathbf{y}' = \mathbf{f}(\mathbf{x}, \mathbf{y}) \\ \mathbf{y}(\mathbf{x}_0) = \mathbf{y}_0 \end{cases} \quad (3-22c)$$

对于高阶常微分方程 $y^{(m)} = f(x, y, y', y'', \dots, y^{(m-1)})$ (3-23a)

及初始条件 $y(a) = y_0, y'(a) = y'_0, \dots, y^{(m-1)}(a) = y_0^{(m-1)}$ (3-23b)

只需引入 $y_1 = y, y_2 = y', \dots, y_m = y^{(m-1)}$

则可将式 (3-23a) 化为一阶常微分方程组

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \dots\dots \\ y_{m-1}' = y_m \\ y_m' = f(x, y_1, y_2, \dots, y_m) \end{cases} \quad (3-24a)$$

而初始条件式 (3-23b) 相应地转化为

$$\begin{cases} y_1(a) = y_0 \\ y_2(a) = y'_0 \\ \dots\dots \\ y_m(a) = y_0^{(m-1)} \end{cases} \quad (3-24b)$$

这样, 高阶 ODE-IVP 问题 (3-23a)、(3-23b) 即化为与 (3-22a)、(3-22b) 形式相同的一阶 ODE-IVPs 问题。

对于高阶常微分方程组, 用上面的降阶方法也可以化为一阶常微分方程组。因此, 高阶微分方程 (组) 的求解问题变为一阶常微分方程组的求解问题。

对于常微分方程 (组) 的边值问题, 降阶方法完全相似。

所以下面将重点介绍一阶常微分方程 (组) 的初值问题和边值问题的数值解法, 并给出一些计算示例。由于化工方面的 ODE 问题很多, 有关应用实例将在第四章中专门介绍。

3.4.1 常微分方程初值问题 (IVP) 的数值解法

在化学工程中, 常微分方程模型主要有一维分布参数过程稳态模型和集中参数过程动态

模型两大类，其中，前者的自变量为一维空间位置 x 或 r 等，式 (3-21) 和式 (3-22c) 正好可以表示这类模型，后者的自变量为时间 t 。

动态模型也是过程自动控制领域的重点研究对象之一，作者认为，MATLAB 对控制领域的关注程度明显高于对许多其他领域的关注程度，因为 MATLAB 在其提供的常微分方程求解器中，以 t 为自变量。但是，不管自变量的符号是什么，解法完全一样。为了与 MATLAB 保持一致，下面常微分方程初值问题的自变量改用 t 。

一阶常微分方程初值问题 (ODE-IVP) 的一般形式为：

$$\begin{cases} \frac{dy}{dt} = f(t, y) \\ y(t_0) = y_0 \end{cases} \quad (3-25)$$

要求在区间 $[t_0, b]$ 上求解。

一阶常微分方程方程组 (ODE-IVPs) 的向量形式为：

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases} \quad (3-26)$$

式 (3-25) 和 (3-26) 形式分别与式 (3-21) 和 (3-22c) 相同，只是自变量名不同而已。

一阶常微分方程 (组) 初值问题包括非刚性问题和刚性问题两大类，多数情况属于非刚性 ODE 问题，但化工领域也有不少的刚性 ODE 问题。MATLAB 提供的非刚性方程积分函数和刚性方程积分函数分别列于表 3-8 和表 3-9 中。

表 3-8 非刚性方程的积分函数

函数名	算法	精度
ode45	四五阶 Runge-Kutta 法	较高
ode23	二阶 Runge-Kutta 法	低
ode113	可变阶 dams-Bashforth-Moulton 法	

表 3-9 刚性方程的积分函数

函数名	算法	精度
ode15s	基于数值差分的可变阶方法(BDFs, Gear)	低~中
ode23s	二阶改进的 Rosenbrock 法	低
ode23t	使用梯形规则	适中
ode23tb	TR-BDF2 (隐式 Runge-Kutta 法)	低

这些积分函数及其算法，一般而言，精度较高者，其速度较慢，反之亦然。目前，计算机性能很好，运行速度极快，除非运算量很大而需要考虑速度以外，一般尽量使用精度较高的算法。此外，上述各 MATLAB 积分函数都是变步长积分，且自动选择步长，计算稳定可靠。其中，最常用的是 ode45() 和 ode23s()。

下面先介绍 ode45() 的调用方法 (其他积分函数的调用方法基本相同，不再另行介绍)，然后给出非刚性方程的例子，最后再简要介绍刚性问题。

函数 ode45() 的用法

$[t, y] = \text{ode45}(@\text{ODEfun}, \text{tspan}, y_0)$

$[t, y] = \text{ode45}(@\text{ODEfun}, \text{tspan}, y_0, \text{options}, p1, p2, \dots)$

输入变量 ODEfun 自定义函数的函数名，该函数定义此微分方程形式 $y' = f(t, y)$ ，函数 $f(t, y)$ 应当返回一列向量；

tspan	tspan 可以是积分限: $tspan = [t0, tf]$, 也可以是表示一些离散点的向量, 如 $tspan = [t0 \ t1 \ t2 \dots tf]$, 这时 <code>ode45()</code> 将计算在这些离散点处的对应 y 值。
y0	状态变量的初始条件向量 (通常是列向量);
options	积分参数选项 (可选), <code>options = []</code> 时表示取默认值。
p1, p2...	直接传递给自定义函数 <code>ODEfun</code> 的已知参数。
输出变量 t	时间向量 (列)
y	状态变量数组, 是时间的函数 (第一列是状态 1, 以此类推)

例如, 如果有三个状态变量, 且时间向量有 30 个元素, 那么状态变量矩阵具有 30 行和三列。

【例 3-28】 求解初值问题:
$$\begin{cases} y' = y - \frac{2x}{y} & (0 \leq x \leq 1) \\ y(0) = 1 \end{cases}$$

程序说明 函数 `f()` 定义微分方程, 主程序分别用四-五阶龙格-库塔法函数 `ode45()` 和二-三阶龙格-库塔法函数 `ode23()` 进行计算。

程序清单 xODE.m

```
function xODE % Demonstrate how to use ode45() and ode23()
clear all; clc, format long
y0 = 1;
[x1, y1] = ode45(@f, [0, 1], y0)
[x2, y2] = ode23(@f, [0, 1], y0)
plot(x1, y1, 'k-', x2, y2, 'b--'), xlabel('x'), ylabel('y')

% -----
function dydx = f(x, y)
dydx = y - 2*x/y;
```

计算结果 采用四-五阶龙格-库塔法和二-三阶龙格-库塔法的结果分别是 $y = 1.73205081676653$ ($x = 1.0$) 和 $y = 1.73215487941780$ ($x = 1.0$), 而已知精确值为 $y = 1.7320508$ ($x = 1.0$)。可见, 四-五阶龙格-库塔法的结果和精确值基本相同, 二-三阶龙格-库塔法的精度稍差一些, 但仍比较满意。

【例 3-29】 求解初值型常微分方程组:
$$\begin{cases} y'_1 = 3y_1 + 2y_2 \\ y'_2 = 4y_1 + y_2 \\ y_1(0) = 0, y_2(0) = 1, 0 \leq x \leq 1 \end{cases}$$

程序说明 用函数 `f()` 定义微分方程组, 并分别用 `ode45()` 和 `ode23()` 积分。

注 在定义微分方程组的函数 `f()` 中, 函数返回值必须是列向量: $dydx = [f1; f2]$, 若为行向量 $dydx = [f1 \ f2]$ 则出错。

程序清单 xODEs.m

```
function xODEs
clear all; clc
```

```

y0 = [0 1];
[x1, y1] = ode45(@f, [0:0.2:1], y0);
[x2, y2] = ode23(@f, [0:0.2:1], y0);
disp('Results by using ode45(): '), disp('      x      y(1)      y(2)'), disp([x1 y1])
disp('Results by using ode23(): '), disp('      x      y(1)      y(2)'), disp([x2 y2])

% -----
function dydx = f(x, y)      % Define simultaneous ODE equations
f1 = 3*y(1) + 2*y(2);
f2 = 4*y(1)+y(2);
dydx = [f1; f2];

```

计算结果

使用 ODE45()			使用 ODE23()		
x	y(1)	y(2)	x	y(1)	y(2)
0	0	1.0000	0	0	1.0000
0.2000	0.6332	1.4519	0.2000	0.6329	1.4516
0.4000	2.2396	2.9099	0.4000	2.2371	2.9075
0.6000	6.5123	7.0611	0.6000	6.5003	7.0491
0.8000	18.0499	18.4993	0.8000	18.0019	18.4512
1.0000	49.3492	49.7170	1.0000	49.1820	49.5499

刚性问题

在化学反应和自动控制等领域经常会遇到一类病态方程组，通常称刚性（stiff）方程组或病态方程组。例如，初值问题

$$\begin{cases} y_1' = -0.01y_1 - 99.99y_2 \\ y_2' = -100y_2 \\ y_1(0) = 2, \quad y_2(0) = 1 \end{cases}$$

一般地，设线性方程组的初值问题为
$$\begin{cases} \mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}(x) \\ \mathbf{y}(x_0) = \mathbf{y}_0 \end{cases} \quad (3-27)$$

其中 \mathbf{A} 是 m 阶方阵，特征值为 $\lambda_1, \lambda_2, \dots, \lambda_m$ ， $\operatorname{Re}\lambda_i < 0$ 。

如果系数矩阵 \mathbf{A} 的特征值 λ_i 具有如下特征：

$$\begin{cases} \operatorname{Re}\lambda_i < 0 \\ \max |\operatorname{Re}\lambda_i| / \min |\operatorname{Re}\lambda_i| \text{ 是个大数} \end{cases} \quad (3-28)$$

则称（3-27）为刚性方程组。其中 $\max |\operatorname{Re}\lambda_i| / \min |\operatorname{Re}\lambda_i|$ 称为刚性比。上例的刚性比为 $100/0.01=10000$ 。

对于非线性方程组的初值问题
$$\begin{cases} \mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \\ \mathbf{y}(x_0) = \mathbf{y}_0 \end{cases} \quad (3-29)$$

也存在同样的问题，只是 λ_i 表示矩阵 $\partial \mathbf{f} / \partial \mathbf{y}$ 的第 i 个特征值，

$$\partial \mathbf{f} / \partial \mathbf{y} = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \dots & \frac{\partial f_1}{\partial y_m} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \dots & \frac{\partial f_2}{\partial y_m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial y_1} & \frac{\partial f_m}{\partial y_2} & \dots & \frac{\partial f_m}{\partial y_m} \end{bmatrix} \quad (3-30)$$

对于刚性 ODE 方程(组), 应该用表 3-8 中的积分函数(如 ode23s)求解。如果考虑程序的通用性, 那么可在程序中先判断 ODE 方程(组)是否为刚性, 然后选择相应的求积函数求解。不过, 若嫌麻烦, 可不必编写此部分判断代码, 一般的做法是, 优先用 ode45()求解, 若能顺利解出即大功告成, 否则, 若长时间没解完(或者感觉像死循环或死机), 可能就是刚性问题, 这时, 可中断程序改用刚性求积函数(如 ode23s)。

【例 3-30】 解刚性常微分方程组

$$\begin{cases} y_1' = 0.04(1 - y_1) - (1 - y_2)y_1 + 0.0001(1 - y_2)^2 \\ y_2' = -10^4 y_1' + 3000(1 - y_2)^2 \\ y_1(0) = 0, y_2(0) = 1, 0 \leq x \leq 100 \end{cases}$$

程序为 xStiffODEs.m (见光盘)。读者可在程序中用注释符关闭或打开有关语句, 分别用 ode45()和 ode23s()计算, 结果将发现用 ode45()导致发散, 而用 ode23s()会很快收敛。

3.4.2 常微分方程边值问题(BVP)的数值解法

BVP 问题的典型例子是一维扩散问题, 它通常是二阶常微分方程。

如前所述, 由于任何高阶常微分方程都可转化为一阶常微分方程组, 因此只需讨论一阶常微分方程组两点边值问题(ODE-BVPs)的求解方法。

标准的一阶常微分方程组两点边值问题:

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, y_2, \dots, y_N) \quad i = 1, 2, \dots, N \quad (3-31a)$$

$$\text{B.C. 在 } x = a, \quad g_{1j}(a, y_1, y_2, \dots, y_N) = 0 \quad j = 1, 2, \dots, n_1 \quad (3-31b)$$

$$\text{在 } x = b, \quad g_{2k}(b, y_1, y_2, \dots, y_N) = 0 \quad k = 1, 2, \dots, n_2 \quad (3-31c)$$

式中, $n_2 = N - n_1$ 。该问题希望得到 N 个一阶 ODE 方程组的解, 使该解满足在起点 $x = a$ 处的 n_1 个边界条件和在终点 $x = b$ 处的其余 $n_2 = N - n_1$ 个边界条件。

上述一阶常微分方程组的 BVP 问题的一般向量形式为:

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}), \quad a < x < b \quad (3-32a)$$

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = 0 \quad (3-32b)$$

如果系统(3-32a)中的方程数是 N , 则(3-32b)中的独立条件数也应该是 N 。

有两个重要问题可化简为由方程(3-31a)~(3-31c)描述的标准边值问题:

(1) **微分方程的特征值问题** (Eigenvalue problem for differential equations)

微分方程特征值问题的一般形式是

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, y_2, \dots, y_N, \lambda) \quad i = 1, 2, \dots, N \quad (3-33)$$

式中, 方程右边与参数 λ 有关, 因此方程必须满足 $(N+1)$ 个边界条件, 而不是 N 个。

一般而言, 对任意的 λ 值, 该问题无解。但对于特殊的 λ 值 (特征值), 它存在一个解。

引入一个新的因变量 $y_{N+1} \equiv \lambda$ (3-34)

及另一个微分方程 $\frac{dy_{N+1}}{dx} = 0$ (3-35)

则方程 (3-33) ~ (3-35) 构成标准的 BVP 问题, 即原问题已简化为标准的 BVP 问题。

下列方程是微分方程特征值问题的一个例子:

$$\begin{cases} y'' + \lambda y = 0 \\ y(0) = 0, y(L) = 0 \end{cases}$$

(2) 自由边界问题 (Free boundary problem)

自由边界问题就是只给定一个边值 $x = a$, 而另一个边值 $x = b$ (这里 b 未知) 需要确定以便使方程 (3-31a) 的解满足总共 $(N+1)$ 个的边界条件。

与上面类似, 也引入一个恒定的因变量:

$$y_{N+1} \equiv b - a \quad (3-36)$$

$$\frac{dy_{N+1}}{dx} = 0 \quad (3-37)$$

再定义一个新的自变量 t : $x - a = t y_{N+1}, 0 \leq t \leq 1$ (3-38)

这样, $(N+1)$ 个微分方程组 dy_i/dt 成为标准的 BVP 形式, 其自变量 t 在区间 $[0, 1]$ 内改变。

常微分方程组的边值问题主要有两大类的数值解法。

① 打靶法 (Shooting Method): 即先把 ODE 边界问题转化为初值问题, 然后按初值问题的解法求解。

② 松弛法 (Relaxation Methods): 主要包括有限差分法 (Finite Difference Methods, FDM) 和配置法 (Collocation Methods)。前者用在网格点上的有限差分方程近似代替 ODEs。

打靶法 (Shooting Method)

打靶法基本思想

先把 ODE-BVPs 的 (3-32a) 式化为 ODE-IVPs:

$$\mathbf{u}' = \mathbf{f}(x, \mathbf{u}) \quad (3-39)$$

$$\text{I.C.} \quad \mathbf{u}(a) = \mathbf{s} \quad (3-40)$$

式中, \mathbf{s} 为未知向量。

现在寻找 \mathbf{s} , 使 $\mathbf{u}(x, \mathbf{s})$ 是 (3-32a) 的解。如果 \mathbf{s} 是

$$\Phi(\mathbf{s}) = \mathbf{g}(\mathbf{s}, \mathbf{u}(b, \mathbf{s})) = \mathbf{0} \quad (3-41)$$

的根, 即满足边界条件方程 (3-32b), 则 $\mathbf{u}(x, \mathbf{s})$ 即为 (3-32a) 的解。

打靶法求解步骤

对于标准的一阶 ODE-IVPs (3-31a) ~ (3-31c), 用打靶法求解时, 从 $a \rightarrow b$ 积分, 使积分结束时满足边界条件。打靶法实际上执行多变量全局收敛, 即对具有 n_2 个变量的 n_2 个函数求零解。打靶法的具体步骤如下:

在起始点 $x = a$, 指定 N 个起始值 y_i , 但有 n_1 个条件约束。因此有 $n_2 = N - n_1$ 个可自由调节的变量。设这些可调数值组成 n_2 维向量 \mathbf{V} 的元素, 则 (3-31b) 变成:

$$y_i(a) = y_i(x; V_1, \dots, V_{n_2}) \quad i = 1, \dots, N \quad (3-42)$$

在终点 $x = b$ 处, 定义一个残差向量 \mathbf{F} , 该向量也具有 n_2 维, 其元素可以测量是否已满

足了 $x = b$ 处的 n_2 个边界条件。最简单地, 只用 (3-31c) 式中的左边:

$$F_k = g_{2k}(b, y) \quad k = 1, \dots, n_2 \quad (3-43)$$

只有当 $x = b$ 处的边界条件满足时, F 才等于 0。

可以用 Newton-Raphson 方法寻找向量 V , 使 $F = 0$ 。当然, 用最优化方法, 也可以搜索向量 V , 使 F 的绝对值最小 (相当于 $F = 0$)。

有关打靶法算法的细节, 可参阅 Davis(1984)^[13]、徐自新 (1990)^[14]。

有限差分法 (FDM)

有限差分方法是用有限差分格式近似代替 ODEs 中的导数项, 从而将 ODEs 离散为线性或非线性代数方程组, 然后再求解此代数方程组。第 5 章将介绍如何使用这种方法来求解 PDE 方程 (组)。详细算法可参考文献^[3, 15]。这里不再叙述。

配置法 (CM)

配置法是加权余量法 (Method of Weighted Residuals) 的一种, 它常用于求解微分方程 (包括 PDE 方程) 边值问题。正交配置法是配置法很有效的一种方法, 第 5 章将介绍如何使用该方法来求解 PDE 方程。详细参阅 Finlayson (1980)^[16]、张建侯和许锡恩 (1989)^[17]。

MATLAB 提供的 ODE-BVPs 求解器, 是采用配置法进行求解的, 其算法非常有效, 几乎可以解决所有的 ODE-BVPs 问题, 下面介绍其使用方法。

MATLAB 的 ODE-BVPs 求解器

MATLAB 的 ODE-BVPs 求解器包含的主要函数有: `bvpinit()`、`bvp4c()` 和 `deval()`。求解步骤:

- 调用函数 `bvpinit()` 以生成在 $[a, b]$ 内的给定初始网格 x 上的初始猜测解 (结构);
- 调用函数 `bvp4c()` 利用配置法得到解 `sol` (结构);
- 调用函数 `deval()` 计算在 $[a, b]$ 内任意点 `xint` 处的解。

下面先介绍这三个函数的一般用法, 然后给出示例。

函数 `bvpinit()` 的用法

功能: 生成一个初始网格上的初始猜测解结构, 以便使用 `bvp4c()` 求解 ODE 的 BVP 问题。

调用格式: `solinit = bvpinit(x, yinit)`

输入参数: x x 为初始网格 (即 `solinit.x`)

$yinit$ 作为猜测解的一个常数向量, 即在 $[a, b]$ 内任意点处的猜测解。

函数 `bvp4c()` 的用法

功能: 用配置法求解常微分方程 (组) 的两点边界值问题, 即式 (3-32a)、(3-32b)。

调用格式: `sol = bvp4c(@ODEfun, @BCfun, solinit)`

`sol = bvp4c(@ODEfun, @BCfun, solinit, options, p1, p2...)`

输入参数:

ODEfun 定义 ODE 方程 (组) 的函数, 该函数有两个参数: $f = \text{ODEfun}(x, y)$ 其中, x 为标量, 而 y 为向量, 返回一个代表 $f(x, y)$ 的列向量。

BCfun 定义边界条件的函数, 该函数有两个参数: `BCfun(ya, yb)`, 其中, ya 和 yb 分别代表左边界 $y(a)$ 和右边界 $y(b)$, 该函数返回一个代表 $bc(y(a), y(b))$ 的列向量。若是方程组, 则 ya 和 yb 是列向量, 其中 $ya(1)$ 、 $yb(1)$ 分别表示第 1 个因变量的左、右边界, $ya(2)$ 、 $yb(2)$ 分别表示第 2 个因变量的左、右边界, 以此类推。

- Solinit** 表示解的初始猜测值的一个结构，该结构有两个域。
x——排序的初始网格节点：Solinit.x(1) = a, ..., Solinit.x(end) = b
y——解的初始猜测值：在节点 Solinit.x(i)处 y(x(i))的初始猜测值为 Solinit.y(:, i)。该结构可先用函数 bvpinit()生成。
- options** 为了大大减少运行时间，使用 options 以提供雅可比和/或使 ODEfun()向量化，详见 BVPSET 和 SHOCKBVP。
- p1, p2...** 已知参数，可传递给函数 ODEfun()和 BCfun()以及 options 指定的所有函数。若 options 取默认值，则有参数 p1, p2...时，令 options = []。

输出参数：

- sol** 返回的解 sol 是一个结构，它包括以下四个域：
sol.x 代表网格节点；
sol.y 为网格节点 sol.x 处的解；
sol.yp 为网格节点 sol.x 处的导数 y'(x)的近似解；
sol.solver 表示求解器，即函数 bvp4c()。

可见，利用函数 bvp4c()可计算得到区间[a,b]内的连续解以及连续的一阶导数。

函数 deval()的用法

功能：计算微分方程 IVP 问题和 BVP 问题在任意点处 xint 的解。

调用格式： yint = deval(sol, xint)

输入参数：

- sol** 由常微分方程 IVP 问题求解器(ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb)、BVP 问题求解器 bvp4c(), 或者延时微分方程 (DDE) 求解器 dde23()等求解得到的一个结构解 sol。它与 bvp4c()中的 sol 相同。
- xint** 在求解区间[sol.x(1) sol.x(end)]内的任意点

【例 3-31】 求解两点边值问题：
$$\begin{cases} y'' - y = 10 \\ y(0) = y(1) = 0 \end{cases}$$

程序说明 记 $y_1 = y$, $y_2 = y_1'$, 则原问题等价于方程组：

$$\begin{cases} y_1' = y_2 \\ y_2' = y_1 + 10 \end{cases} \quad (1)$$

$$\text{B.C. (边界条件):} \quad y_1(0) = 0, y_1(1) = 0 \quad (2)$$

程序中，函数 ODEfun()定义方程组 (1)，注意该函数必须返回列向量（与 ode45 和 ode23 等要求一样）。BCfun()定义边界条件 (2)，ya(1)、yb(1)分别表示因变量 y_1 的左、右边界。

程序清单 xBVPl.m

```
function xBVP1
clear all; clc, format long e, a = 0; b = 1;

% solution is obtained using an initial guess of y1(x)=1, y2(x)=0
solinit = bvpinit(linspace(a, b, 10), [1 0]);
sol = bvp4c(@ODEfun, @BCfun, solinit);
x = [0:0.1:0.5];
```

```

y = deval(sol, x);
DispResult_xBVP           % 调用 DispResult_xBVP.m (见光盘) 以显示结果

% -----
function dydx = ODEfun(x, y)
dydx = [y(2); y(1)+10];

% -----
function bc = BCfun(ya, yb)
bc = [ya(1); yb(1)];

```

计算结果 计算结果及精确解列于下表中作比较。

x	0	0.1	0.2	0.3	0.4	0.5
精确解	0	-0.412846057	-0.729740656	-0.95385538	-1.08743325	-1.13181116
数值解	0	-0.412846391	0.729741888	-0.953857557	1.08743610	-1.13181425

注: $x = 0.6, 0.7, 0.8, 0.9, 1$ 的 y 值分别与 $x = 0.4, 0.3, 0.2, 0.1, 0$ 的 y 值相等。

【例 3-32】 求解两点边值问题:
$$\begin{cases} y'' - (1+x^2)y = 1 \\ y(0) = 1, y(1) = 3 \end{cases}$$

程序说明 记 $y_1 = y$, $y_2 = y_1'$, 则原问题等价于方程组:
$$\begin{cases} y_1' = y_2 \\ y_2' = (1+x^2)y_1 + 1 \end{cases} \quad (1)$$

B.C. (边界条件) 应变换为以下标准形式:
$$\begin{cases} y_1(0) - 1 = 0 \\ y_1(1) - 3 = 0 \end{cases} \quad (2)$$

程序中, 函数 ODEfun() 定义方程组 (1), BCfun() 定义边界条件 (2)。

程序清单 xBVP2.m

```

function xBVP2
clear all; clc, a = 0; b = 1;

% solution is obtained using an initial guess of y1(x) = 0, y2(x) = 0
solinit = bvpinit(linspace(a, b, 10), [0 0]);
sol = bvp4c(@ODEfun, @BCfun, solinit);
x = [0:0.1:1];
y = deval(sol, x);
DispResult_xBVP           % 调用 DispResult_xBVP.m 以显示结果

% -----
function dydx = ODEfun(x, y)
dydx = [y(2); (1+x^2)*y(1)+1];

% -----
function bc = BCfun(ya, yb)
bc = [ya(1)-1; yb(1)-3];

```

计算结果

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	1.0000	1.0743	1.1695	1.2869	1.4284	1.5965	1.7947	2.0274	2.3004	2.6214	3.0000

有未知参数的 BVP 问题

有些边值问题在计算 $y(x)$ 时需要同时计算未知参数 p 向量:

$$y' = f(x, y, p), \quad a < x < b \quad (3-44a)$$

$$g(y(a), y(b), p) = 0 \quad (3-44b)$$

对此问题, 用域 `solinit.parameters` 提供未知参数的猜测值。求解此 BVP 问题时, 先给定初始参数值, 然后重复求解 BVP 得到所需的参数。返回的参数为 `sol.Parameters`, 包含一个参数值集的问题解 `sol` 可用作另一参数集的初始猜测解 `solinit`。

若有未知参数, 则用下面命令得到包括未知参数猜测向量 `params` 在内的猜测值。

`solinit = bvpinit(x, yinit, params)`

奇异 BVP 问题

函数 `bvp4c()` 可求解一类奇异 BVP 问题, 包括有未知参数 p 的 BVP 问题:

$$\begin{aligned} y' &= S * y / x + f(x, y, p) \\ g(y(0), y(b), p) &= 0 \end{aligned} \quad (3-45)$$

求解区间为 $[0, b]$, 其中 $b > 0$ 。

通常, 柱形对称或球形对称的 PDE 问题的求解可以转变为此类 ODE 问题的求解。对于奇异问题, 可将常数矩阵 S 作为 BVPSET 的 'SingularTerm' 选项的赋值, 且 `ODEfun()` 只估算 $f(x, y, p)$ 。边界条件必须与必要条件 $S * y(0) = 0$ 相符, 而且初始猜测值应该满足此条件。

3.5 小结

插值与拟合

插值: `spline()`

拟合: 多项式拟合——`polyfit()`, `polyval()`

最小二乘样条拟合——`csaps()`、`spap2()`、`spaps()`

用于计算函数值的函数: `polyval()`、`fval()`

数值积分与数值微分

数值积分 `quad`——自适应 Simpson 求积公式 (低阶)

`quadl`——自适应 Lobatto 求积公式 (高阶)

数值微分

① 最小二乘多项式拟合法 — `polyfit()`, `polyder()`, `polyval()`: 离散数据 $\xrightarrow{\text{polyfit()}}$ 向量 p 表示的多项式拟合函数 $\xrightarrow{\text{polyder()}}$ 导函数 $pp \xrightarrow{\text{polyval()}}$ pp 在 xi 处的导数值。

② 最小二乘样条拟合法 — `csaps()` 或 `spap2()` 或 `spaps()`, `fnder()`, `fval()`: 离散数据 $\xrightarrow{\text{csaps() 或 spap2() 或 spaps()}}$ 样条拟合函数 $sp \xrightarrow{\text{fnder()}}$ sp 的导数 $pp \xrightarrow{\text{fval()}}$ pp 在 xi 处的导数值。

代数方程 (组) 的数值解法

线性代数方程组的解法: 用 `inv()` 或左除 “\”

已知 $AX = b$, 则 $X = \text{inv}(A) * b$ 或 $X = A \backslash b$

非线性代数方程 (组) 的解法

单变量非线性方程: fzero()和 roots()。

多变量非线性系统(方程组): fsolve()

常微分方程(组)的数值解法

ODE-IVP 问题: 非刚性方程(组) — ode45(), 刚性方程(组) — ode23s()

ODE-BVP 问题: bvpinit()、bvp4c()

习 题

3-1 热力学实验测得关于氧气的压缩因子数据如下。

p/atm	0.0	20.0	40.0	60.0	80.0	100.0	120.0
z	1.0	0.98654	0.97420	0.96297	0.95286	0.94387	0.93599

试按逸度系数的定义式 $\ln \phi = \int_0^p \frac{z-1}{p} dp$ 计算其逸度系数。

3-2 已知函数 $y = e^x$ 的下列数值:

x	2.5	2.6	2.7	2.8	2.9
y	12.1825	13.4637	14.8797	16.4446	18.1741

试计算 $x = 2.7$ 处函数的一、二阶导数值, 并与真实值比较。

3-3 求解初值问题: $\begin{cases} y' = x^2 + y^2 \\ y(0) = 0 \end{cases} \quad (0 \leq x \leq 1)$

3-4 已知下面 ODE 方程组:

$$\begin{aligned} \frac{dy_1}{dt} &= -0.013y_1 - 1000y_1y_3 \\ \frac{dy_2}{dt} &= -2500y_2y_3 \\ \frac{dy_3}{dt} &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3 \end{aligned}$$

初始条件为 $y_1(0) = 1$, $y_2(0) = 1$, $y_3(0) = 0$, 试从 $t_0 = 0$ 到 $t_f = 50$ 进行数值积分。

3-5 求解下列边值问题, 并与精确解比较。

- (a) $\begin{cases} y'' - 4y, & 0 < x < 1 \\ y(0) = 1, & y(1) = e^2 \end{cases} \quad (\text{精确解 } y = e^{2x})$
- (b) $\begin{cases} y'' + (x+1)y' - 2y = (1-x^2)e^{-x}, & 0 < x < 1 \\ y'(0) = 2, & y'(1) = 1/e \end{cases} \quad (\text{精确解 } y = (x-1)e^{-x})$
- (c) $\begin{cases} y'' + y'^2 - \frac{1}{x}y = \frac{1}{x}(2 - \ln x), & 1 < x < 2 \\ 2y(1) - y'(1) = 0, & y'(2) = 3/2 \end{cases} \quad (\text{精确解 } y = x + \ln x)$

参 考 文 献

- 王洪然编著. MATLAB 5.X 与科学计算. 北京: 清华大学出版社, 2000
- 苏金明, 阮沈勇编著. MATLAB 6.1 实用指南(上册). 北京: 电子工业出版社, 2002
- A. Constantinides and N. Mostoufi, Numerical Methods for Chemical Engineers with MATLAB Applications. Prentice Hall PRT, Upper Saddle River, New Jersey, 1999
- 程正兴, 李永根. 数值逼近与常微分方程数值解. 西安: 西安交通大学出版社, 2000
- 张吉瑞编著. 化工数值方法. 北京: 中国石化出版社, 1995, 106, 119, 124

- 6 Octave Levenspiel, Chemical Reaction Engineering. Third Edition. John Wiley & Sons, Inc., 1999, 104, 60)
- 7 周爱月. 化工数学. 北京: 化学工业出版社, 1993, 149, 161, 165
- 8 Ricardson J. F. and Peacock D. G., Chemical Engineering, Vol.3, Chemical & Biochemical Reactor & Process Control, 3rd edition. 世界图书出版社, 1994
- 9 Raman R., Chemical Process Computation. London: Elsevier Applied Science Publishers. 1985. 中译本. 许锡恩, 张福芝, 王保国, 吴诗华译. 化工过程计算. 北京: 化学工业出版社, 1992, 209, 122
- 10 王树森编. 化学工程计算方法. 北京: 化学工业出版社, 1989. 参考原文: 日本化学工学协会编. 麻德贤译. 化学工程程序设计例题与习题集. 化学工业出版社, 1984. 21
- 11 倪进方. 化工过程设计. 化学工业出版社, 1999, 118
- 12 William L. Luyben, Process modeling, simulation, and control for chemical engineers, Second edition. McGraw-Hill, Inc., 1990, 46, 124
- 13 Mark E. Davis, Numerical Methods and Modeling for Chemical Engineers, John Wiley & Sons, Inc., 1984
- 14 徐自新编著. 微分方程近似解. 上海: 华东化工学院出版社, 1990, 16~17, 85, 95
- 15 Press W. H., Teukdsky S. A., Vetterling W. T., Flannery B. P., Numerical Recipes in C, Cambridge University Press, Cambridge UK, 1992
- 16 Finlayson Bruce A. Nonlinear Analysis in Chemical Engineering. New York: McGraw-Hill, 1980
- 17 张建侯, 许锡恩编著. 化工过程分析与计算机模拟. 北京: 化学工业出版社, 1989
- 18 MATLAB Help, Version 6.5, Release 13, The Mathworks, Inc., 2002

第4章 化工中的常微分方程及其求解

在科学研究与工程实践中最常遇到的数学模型是以微分方程(组)的形式出现的。第3章已经介绍了常微分方程(组)的一般求解方法,本章将介绍具体的化工应用实例。

在化学工程中,常微分方程模型主要有一维分布参数过程稳态模型和集中参数过程动态模型两大类,其中,前者自变量为一维空间位置 x 或 r 等,对于非等温、非绝热反应器的一维稳态模拟常导致一组非线性的 ODE-IVP 模型;后者自变量为时间 t ,主要用于开、停车非稳态过程研究、间歇过程分析、过程控制系统设计和分析以及操作人员的培训等。对于扩散-反应系统,主要是 ODE-BVP 问题。下面详细介绍间歇反应器、CSTR、管式反应器、半连续反应器、传质过程、伴有反应的扩散过程、传热过程(化学热泵)、流体流动、生化反应、膜反应器以及过程控制等应用例子。

4.1 间歇反应器 (Batch Reactor)

【例 4-1】 间歇反应器中的连串-平行复杂反应系统^[1]

在间歇反应器中进行液相反应制备产物 B, 反应网络如图 4-1 所示。

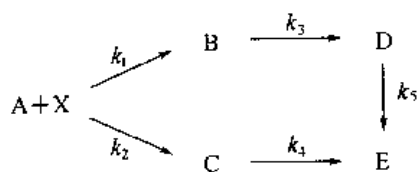


图 4-1 复杂反应网络

反应可在 180~260℃ 的温度范围内进行, 反应物 X 大量过剩, 而 C、D 和 E 为副产物。各反应均为一级动力学关系: $r = -kC$, 式中 $k = k_0 e^{-\frac{E_a}{RT}}$ 。已知参数: $k_{01}=5.78052 \times 10^{10}$, $k_{02}=3.92317 \times 10^{12}$, $k_{03}=1.64254 \times 10^4$, $k_{04}=6.264 \times 10^8$, $k_{05}=2.1667 \times 10^{-4}$, $E_{a1}=124670$, $E_{a2}=150386$, $E_{a3}=77954$, $E_{a4}=111528$ 。初始浓度为: $t=0$, $C_A=1 \text{ kmol/m}^3$, $C_C=C_B=C_D=C_E=0 \text{ kmol/m}^3$ 。

用最优化方法求得使产物 B 收率最大的最优反应温度为 224.6℃ (见 6.3 例 [6-15]), 试计算 (1) 在此最优反应温度下各组分浓度随时间的动态变化; (2) 最优反应时间。

数学模型 根据各组分的物料平衡和动力学方程容易导出以下模型方程组:

$$\frac{dC_A}{dt} = -(k_1 + k_2)C_A \quad (1)$$

$$\frac{dC_B}{dt} = k_1 C_A - k_3 C_B \quad (2)$$

$$\frac{dC_C}{dt} = k_2 C_A - k_4 C_C \quad (3)$$

$$\frac{dC_D}{dt} = k_3 C_B - k_5 C_D \quad (4)$$

$$\frac{dC_E}{dt} = k_4 C_C + k_5 C_D \quad (5)$$

常微分方程组 (1) ~ (5) 的初始条件已给定, 因此可进行数值求解。

程序说明 用 MATLAB 四-五阶龙格-库塔法函数 ode45() 求解该常微分方程组, 得到各组分的浓度变化曲线或动态数据。其中, MassEquations() 定义 ODE 方程组 (1) ~ (5)。对计

算得到的动态离散数据 C_B-t ，直接用 $\max()$ 求取最大 C_B 值及对应的最优反应时间。

程序清单 *BatchReactor.m*

```
function BatchReactor
clear all; clc
T = 224.6 + 273.15; R = 8.31434; % T: 反应器温度(K), R:理想气体常数(kJ/kmol K)
k0 = [5.78052E+10 3.92317E+12 1.64254E+4 6.264E+8]; % 指前因子, 1/s
Ea = [124670 150386 77954 111528]; % 活化能, kJ/kmol

C0 = [1 0 0 0 0]; tspan = [0 1e4]; % C0: 初始浓度 C0(i), kmol/m^3
[t, C] = ode45(@MassEquations, tspan, C0, [], k0, Ea, R, T)

% 绘图
plot(t, C(:,1), 'r-', t, C(:,2), 'k:', t, C(:,3), 'b-', t, C(:,4), 'k--');
xlabel('Time (s)'); ylabel('Concentration (kmol/m^3)'); legend('A', 'B', 'C', 'D')
CBmax = max(C(:,2)); % CBmax: the maximum concentration of B, kmol/m^3
yBmax = CBmax/C0(1) % yBmax: the maximum yield of B
index = find(C(:,2)==CBmax);
t_opt = t(index) % t_opt: the optimum batch time, s

% -----
function dCdt = MassEquations(t, C, k0, Ea, R, T)
k = k0.*exp(-Ea/(R*T)); k(5) = 2.16667E-04; % Reaction rate constants, 1/s

% Reaction rates, kmol/m^3 s
rA = -(k(1)+k(2))*C(1); rB = k(1)*C(1)-k(3)*C(2); rC = k(2)*C(1)-k(4)*C(3);
rD = k(3)*C(2)-k(5)*C(4); rE = k(4)*C(3)+k(5)*C(4);

dCdt = [rA; rB; rC; rD; rE]; % Mass balances
```

计算结果 在 224.6℃ 反应温度下各组分浓度的动态变化如图 4-2 所示。求得最优反应时间 $t_{\text{opt}} = 740$ s，相应的产物 B 最高收率为 0.8129。

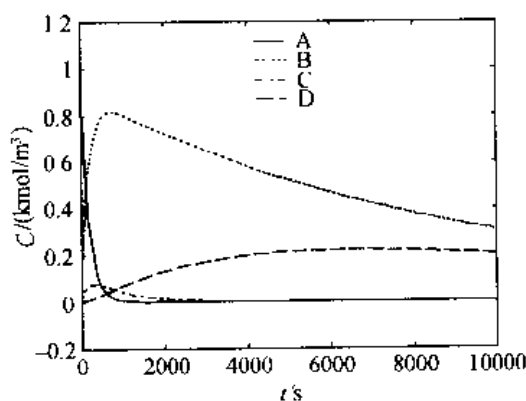


图 4-2 224.6℃ 下各组分浓度随时间的变化曲线

符号说明

C	浓度	kmol/m^3	t	反应时间	s
E_a	活化能	kJ/kmol	T	反应器温度	K
k	反应速率常数	$1/\text{s}$	k_0	指前因子	$1/\text{s}$
R	气体常数	$\text{kJ}/(\text{kmol}\cdot\text{K})$			

下标

A、B、C、D 和 E 组分名称

1、2、3、4 和 5 第 1、2、3、4 和第 5 个反应式

4.2 连续槽式搅拌反应器 (CSTR)

【例 4-2】连续槽式搅拌反应器 (CSTR) 开车过程的动态模拟及稳态模拟^[1]

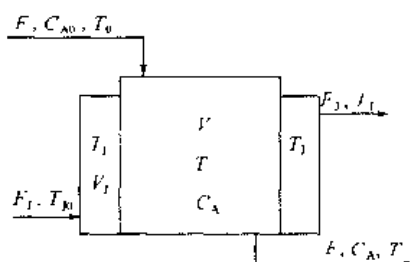


图 4-3 非等温 CSTR

在一个有夹套冷却的 CSTR 反应器 (图 4-3) 中进行一级放热反应: $A \rightarrow P + \Delta H_r$, 其动力学方程为:

$$-r_A = kC_A, \quad k = k_0 \exp\left(-\frac{E}{RT}\right) \quad (1)$$

(a) 模拟计算开车时反应器的动态行为 (动态模拟), 并绘制 CSTR 反应器的相平面图。

(b) 计算反应器达到稳态时的状态 (稳态模拟)。

数学模型

质量衡算方程
$$V \frac{dC_A}{dt} = FC_{A0} - FC_A - kVC_A$$

即
$$\frac{dC_A}{dt} = \frac{C_{A0} - C_A}{\tau} - kC_A \quad (2)$$

式中,
$$\tau = \frac{V}{F} \quad (3)$$

热量衡算方程
$$V\rho c_p \frac{dT}{dt} = Q_g - Q_r$$

即
$$\frac{dT}{dt} = \frac{Q_g - Q_r}{V\rho c_p} \quad (4)$$

式中, Q_g 为反应的放热量; Q_r 为移走的热量。

$$Q_g = kC_A V(-\Delta H), \quad Q_r = F\rho c_p(T - T_0) + UA(T - T_j) \quad (5)$$

因此, 式 (2) 和 (4) 组成动态模型。

令 $dC_A/dt = 0$ 和 $dT/dt = 0$, 则由式 (2) 和 (4) 得到稳态模型:

$$\frac{C_{A0} - C_A}{\tau} - kC_A = 0 \quad (6)$$

$$Q_g - Q_r = 0 \quad (7)$$

已知: $F = 10^{-8} \text{ m}^3/\text{s}$, $C_{A0} = 5.0 \text{ kmol/m}^3$, $T_0 = 300 \text{ K}$, $T = 250 \text{ K}$, $T_j = 305 \text{ K}$, $V = 2.0 \times 10^{-6} \text{ m}^3$, $\rho = 1000 \text{ kg/m}^3$, $c_p = 4.187 \text{ kJ}/(\text{kg}\cdot\text{K})$, $UA = 5.68 \times 10^{-6} \text{ kJ}/(\text{K}\cdot\text{s})$, $H_r = -4.19 \times 10^4 \text{ kJ/kmol}$, $k_0 = 8.03 \times 10^{12} \text{ 1/s}$, $E = 9.42 \times 10^4 \text{ kJ/kmol}$, $R = 8.317 \text{ kJ}/(\text{kmol}\cdot\text{K})$ 。

程序说明 先以函数 DynamicModel() 定义 ODE 方程组 (2) 和 (4), 再调用 ode45() 求解在时

间 $t_{span} = [0 \ 5000]$ 上的浓度和温度动态数据。用循环语句可以在一定范围内改变浓度和温度的初值，并计算不同初值下 CSTR 的动态数据，由此可绘出 CSTR 反应器的相平面图。

程序清单 见光盘中的 *CSTR.m*。

计算结果 CSTR 反应器中的 $Q_g(Q_r)$ - T 关系如图 4-4。CSTR 反应器的相平面图如图 4-5 所示，由图可见，该 CSTR 反应器只有一个定态点。

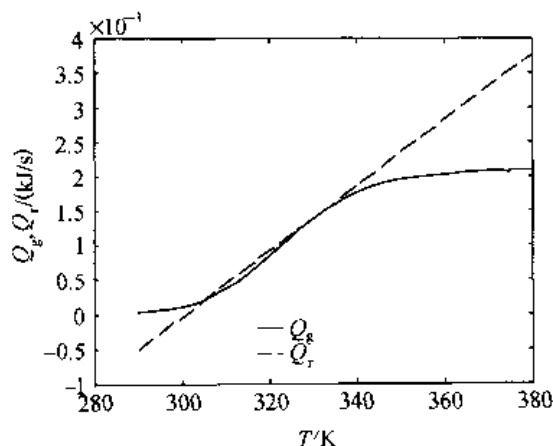


图 4-4 CSTR 反应器中的 $Q_g(Q_r)$ - T 的关系

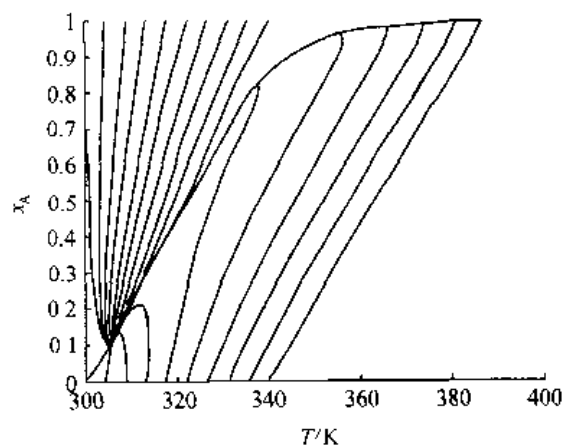


图 4-5 CSTR 反应器的相平面图

【例 4-3】三个串联的 CSTR 等温反应器^[2]

在如图 4-6 所示的三个串联 CSTR 等温反应系统中，发生简单一级反应： $A \xrightarrow{k} B$ ，

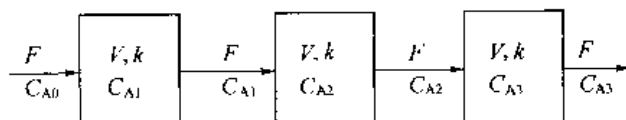


图 4-6 三个串联的 CSTR 反应系统

已知初始条件及参数为： $C_{A0} = 1.8 \text{ kmol/m}^3$ ， $C_{A10} = 0.4 \text{ kmol/m}^3$ ， $C_{A20} = 0.2 \text{ kmol/m}^3$ ， $C_{A30} = 0.1 \text{ kmol/m}^3$ ， $k = 0.5 \text{ min}^{-1}$ ， $\tau = 2 \text{ min}$ 。试求解三个反应器中组分 A 浓度随时间的变化规律。

数学模型 根据物料衡算及反应动力学得到如下模型方程：

$$\frac{dC_{A1}}{dt} = \frac{C_{A0} - C_{A1}}{\tau} - kC_{A1} \quad (1)$$

$$\frac{dC_{A2}}{dt} = \frac{C_{A1} - C_{A2}}{\tau} - kC_{A2} \quad (2)$$

$$\frac{dC_{A3}}{dt} = \frac{C_{A2} - C_{A3}}{\tau} - kC_{A3} \quad (3)$$

程序说明 用 `ode45()` 求解 ODE 方程组 (1) ~ (3)，该方程组由函数 `Equations()` 定义，其中因变量向量 $y = [CA1 \ CA2 \ CA3]$ 。

程序清单 *IsothermCSTRs.m*

```
function IsothermCSTRs % 三个串联的 CSTR 等温反应器
```

```

clear all; clc
CA0 = 1.8; CA10 = 0.4; CA20 = 0.2; CA30 = 0.1; % kmol/m^3
k = 0.5; tau = 2; stoptime = 2.9; % k: 1/min, stoptime: min
[t, y] = ode45(@Equations, [0 stoptime], [CA10 CA20 CA30], [], k, CA0, tau);
disp('    t        CA1        CA2        CA3'), disp([t, y])
plot(t, y(:,1), 'k--', t, y(:,2), 'b:', t, y(:,3), 'r-')
legend('CA_1', 'CA_2', 'CA_3'), xlabel('Time (min)'), ylabel('Concentration')

% -----
function dydt = Equations(t, y, k, CA0, tau)
CA1 = y(1); CA2 = y(2); CA3 = y(3);
dCA1dt = (CA0-CA1)/tau - k*CA1;
dCA2dt = (CA1-CA2)/tau - k*CA2;
dCA3dt = (CA2-CA3)/tau - k*CA3;
dydt = [dCA1dt; dCA2dt; dCA3dt];

```

计算结果 如图 4-7, 在进口浓度 C_{A0} 阶跃增加后, 浓度 C_{A1} 、 C_{A2} 、 C_{A3} 均随时间逐渐增加。

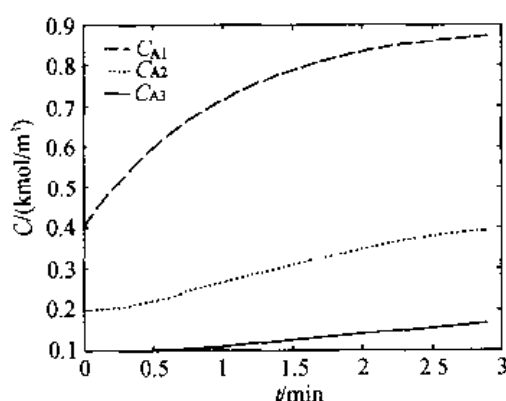


图 4-7 三个反应器中组分 A 的浓度随时间的变化

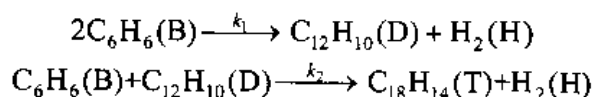
符号说明

C_{A0}	进入第 1 个反应器组分 A 的进口浓度	kmol/m ³
C_{A1} 、 C_{A2} 、 C_{A3}	反应器 1、2、3 中组分 A 的浓度	kmol/m ³
C_{A10} 、 C_{A20} 、 C_{A30}	反应器 1、2、3 中组分 A 的初始浓度	kmol/m ³
k	反应速率常数	min ⁻¹
τ	平均停留时间	min
下标		
0	初始状态	A 反应物 A

4.3 管式反应器

【例 4-4】等温管式反应器^[3]

常压下一理想等温管式反应器中进行苯 (B) 的气相脱氢制二苯基 (D), 反应如下。



已知动力学方程 $r_1 = -k_1(p_B^2 - \frac{p_D p_H}{K_{eq1}})$ lbmol/ft³·h (已反应的苯) (1)

$r_2 = -k_2(p_B p_H - \frac{p_T p_H}{K_{eq2}})$ lbmol/ft³·h (已反应的二苯基) (2)

式中 $k_1 = 1.496 \times 10^7 \times \exp(-\frac{8440}{T})$ (3)

$k_2 = 8.67 \times 10^6 \times \exp(-\frac{8440}{T})$ (4)

试设计一管式反应器，它在 101.325 kPa(1 atm)和 573.9 K 下操作。

- (a) 确定苯转化为二苯基和三苯基的总转化率和各组分分压随 V/F_{A0} 的变化规律；
- (b) 确定（纯苯）进料速率为 $F = 128.2$ lbmol/h、总转化率为 42% 时所需的反应器体积；
- (c) 在什么空速下，可使二苯基的浓度达最大值？
- (d) 研究操作压力对反应的影响（假设等温操作）。

数学模型 以苯为基准的物料平衡方程（稳态）为：

$$\tau = \frac{V_R}{v_0} = C_{A0} \frac{V_R}{F_{A0}} = C_{A0} \int_0^{x_{Ar}} \frac{1}{(-r_A)} dx_A$$

$$\frac{V}{F_{A0}} = -\int \frac{dx_1}{r_1}, \quad \frac{V}{F_{A0}} = -\int \frac{dx_2}{r_2}$$

对于气体加料，通常空速是以度量条件下或标准状态(273K 及 101kPa)下的体积而不是反应器操作温度与压力下的体积来表示。或者说，空速通常以度量条件或标准状态为基准。这里以标准状态为基准，则上式中体积流率 v_0 即基于标准状态，故初始浓度 $C_{A0}=1/22.4$ mol/L（程序中换算为 lbmol/ft³，本例其他变量也保留原非 SI 单位）。

记 $t = \frac{V}{F_{A0}}$ ，并把以上两个方程变为微分形式

$$\frac{dx_1}{dt} = -r_1 \quad (5)$$

$$\frac{dx_2}{dt} = -r_2 \quad (6)$$

由于气体在低压下可视为理想气体，可推导得各组分分压如下：

$$p_B = p(1 - x_1 - x_2) \quad (7)$$

$$p_D = p\left(\frac{x_1}{2} - x_2\right) \quad (8)$$

$$p_H = p\left(\frac{x_1}{2} + x_2\right) \quad (9)$$

$$p_T = p x_2 \quad (10)$$

这样，在反应温度 T 下，由式 (7) ~ 式 (10) 及式 (3)、式 (4) 代入式 (1)、式 (2) 可知， $r_1 = r_1(x_1, x_2)$ 、 $r_2 = r_2(x_1, x_2)$ ，因此，方程组 (5) ~ (6) 只有 x_1 和 x_2 两个未知变量，可求出数值解。

由于等温，所以平衡常数 K_{eq1} 和 K_{eq2} 为固定值。在反应温度 $T = 573.9$ 下，平衡常数为：
 $K_{eq1} = 0.312$, $K_{eq2} = 0.480$ 。

程序说明 用 ode45() 求解 ODE 方程组 (5)、(6)。在自定义函数 Euqations() 中，先计算式 (7) ~ (10)，再计算式 (1) 和 (2)，最后计算 ODE 方程 (5)、(6)，而反应速度常数 k_1 、 k_2 在主程序中计算，参数 $keq1$ 、 $keq2$ 、 k_1 、 k_2 和 P 通过被定义为全局变量 (global) 在主程序与函数 Euqations() 之间共享。

程序清单 IsothermTR.m

```
function IsothermTR          % 等温管式反应器 (苯脱氢反应)
clear all; clc
global keq1 keq2 k1 k2 P
stoptime = 1; dt = 0.01;          % stoptime: 空时(space time)
T = 573.9; keq1 = 0.312; keq2 = 0.480; % 反应温度(K), keq1 和 keq2 为平衡常数
k1 = 1.496e7 * exp(-8.44e3/T); k2 = 8.67e6 * exp(-8.44e3/T); % 反应速度常数, 1/atm^2 h
P = 1; F = 128.2;          % 压力(atm), F: 纯苯的进料流量( lb.mol/h)
CA0 = 1/22.4;              % 标准状态下的进料浓度, mol/L
CA0 = CA0/453.6/(1e-3*3.2808^3); % lbmol/ft^3 (1 mol = 1/453.6 lbmol, 1 m = 3.2808 ft)
v0 = FA0/CA0;              % 标准状态下的进料体积流量, ft^3/h
x0 = [0 0]; t = [0, 0.005, 0.01, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.18, 0.22, 0.26, 0.30, 0.40];
% x0: 初始转化率; t: V/FA0, 单位为 ft^3.h/lbmol
[t, x] = ode45(@Euqations, t, x0); % 解 ODE 方程组
tau = CA0*t; SV = 1/tau;          % 空时, h; 空速, 1/h
xBt = x(:,1) + x(:,2);          % xBt: 苯转化为二苯基和三苯基的总转化率
t42p = spline(xBt, t, 0.42);     % 总转化率为 42% 时的(VR/FA0)
VR = t42p*FA0;                  % 总转化率为 42% 时所需的反应器体积, ft^3
PB = 1-x(:,1)-x(:,2); PD = x(:,1)/2-x(:,2); PT = x(:,2); % 苯、二苯基和三苯基的分压
tmax = t(find(PD==max(PD)));      % 使二苯基的分压(浓度)最大所对应的空时
tau_max = tau(find(PD==max(PD))); % 使二苯基的分压(浓度)最大所对应的空时
SV_max = SV(find(PD==max(PD)));   % 使二苯基的分压(浓度)最大所对应的空速

% 输出结果
disp(' Results:')
fprintf('\n 所需的反应器体积为: %.1f %s\n\n', V, 'ft^3 (苯的总转化率为 42% 时)')
fprintf(' 在 V/FA0 为 %.3f %s\n\n', tmax, 'ft^3-h/lbmol 下, 可使二苯基的浓度达最大值')
disp('      t          x1          x2          xBt          PB          PD          PT')
disp([t x(:,1:2) xBt PB PD PT]), plot(t, xBt), xlabel('V/F_A_0, ft^3-hr/lbmol'), ylabel('x_B_t')
figure, plot(t, PB, 'b--', t, PD, 'm-', t, PT, 'r-'), xlabel('V/F_A_0, ft^3-hr/lbmol'), ylabel('P, atm')
legend('P_B', 'P_D', 'P_T')

% -----
function dxdt = Euqations(t, x) % here t = V/FA0
```

global keq1 keq2 k1 k2 P

PB = P*(1-x(1)-x(2)); PD = P*(x(1)/2-x(2)); PH = P*(x(1)/2+x(2)); PT = P*x(2); % 分压, atm

r1 = k1*(PB*PB-PD*PH/keq1); r2 = k2*(PB*PD-PH*PT/keq2); % 反应速度, 1/h

% 物料平衡方程

dx1dt = r1; dx2dt = r2;

dxdt = [dx1dt; dx2dt];

计算结果

(a) 苯转化为二苯基和三苯基的总转化率随 V/F_{A0} 的变化规律如图 4-8 所示。由图可见, 苯的总转化率达到 42% 以后, 再增加空时, 总转化率提高不再明显。所以计算反应器体积时, 总转化率可取 42%。各组分分压随空时的变化如图 4-9 所示, 可见, 二苯基存在最大浓度值。

(b) 苯进料速率为 $F = 128.2 \text{ lbmol/h}$ 、总转化率为 42% 时所需的反应器体积为: 0.447 m^3 (15.8 ft^3);

(c) 在空速为 1630.9 h^{-1} (对应 V/F_{A0} 为 $0.220 \text{ ft}^3 \cdot \text{h/lbmol}$) 下, 可使二苯基的浓度达最大值。

注 以上是压力为 1 atm 下的结果, 改变不同的压力, 重新计算, 即可考察不同操作压力对反应的影响, 读者可稍稍修改程序, 即可实现此功能, 这里从略。

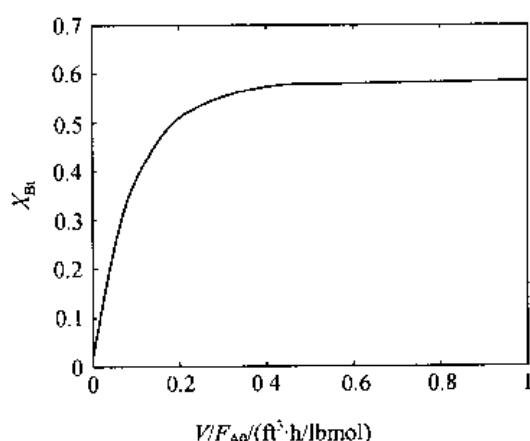


图 4-8 苯的总转化率随空时的变化

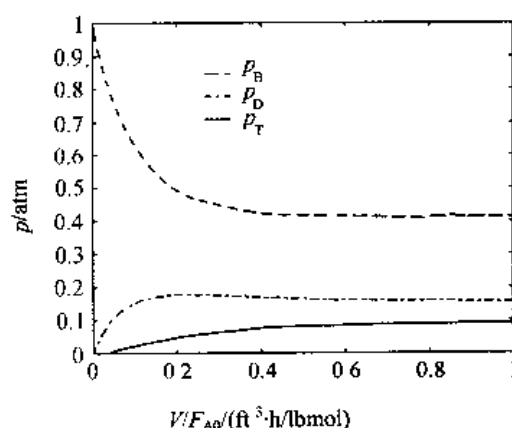


图 4-9 各组分分压随空时的变化

符号说明

C_{A0} 纯苯的进料浓度, lbmol/ft^3

F_{A0} 纯苯的进料流量, lbmol/h

K_{eq1} 、 K_{eq2} 平衡常数

k_1 、 k_2 反应速率常数, $1/(\text{atm}^2 \cdot \text{h})$

p_B 苯的分压, atm

p_D 二苯基的分压, atm

p_H 氢气的分压, atm

p_H 三苯基的分压, atm

r_1 、 r_2 反应速率, $1/h$

T 反应温度, K

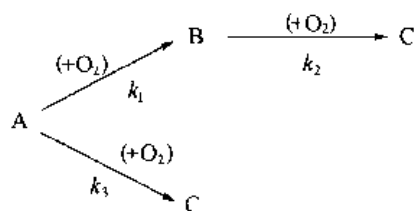
x_1 、 x_2 第 1、2 反应式苯的转化率

V 、 V_R 体积, ft^3

τ 反应器空间, h

【例 4-5】非等温管式反应器——固定床反应器——维稳态拟均相模型的模拟计算 ^[4]

在一列管反应器中进行邻二甲苯(A)氧化制邻苯二酸酐(B), 反应为连串-平行反应:



其中, C 是归并在一起的最终氧化产物 CO 和 CO₂。已知气体混合物的表观质量流速为 $G = 4684 \text{ kg}/(\text{m}^2 \cdot \text{h})$, 催化剂堆积密度 $\rho_B = 1300 \text{ kg}/\text{m}^3$, 气体的平均摩尔质量 $M_m = 29.48 \text{ kg}/\text{kmol}$, 入口处邻二甲苯的摩尔分数 $y_{A0} = 0.00924$, 入口处氧的摩尔分数 $y_0 = 0.208$, 比热容 $c_p = 1.047 \text{ kJ}/(\text{kmol} \cdot \text{K})$, 传热系数 $U = 345.686 \text{ kJ}/(\text{m}^2 \cdot \text{h} \cdot \text{K})$, 管径 $D_t = 0.0254 \text{ m}$, 夹套冷却温度 $T_j = 630 \text{ K}$, 入口温度 $T_0 = 630 \text{ K}$ 。反应式 $A \longrightarrow B$ 的反应热 $H_1 = -1.285 \times 10^6 \text{ kJ}/\text{kmol}$, $A \longrightarrow C$ 的反应热 $H_3 = -4.564 \times 10^6 \text{ kJ}/\text{kmol}$, $A \longrightarrow B$ 的活化能 $E_1 = 1.1304 \times 10^5 \text{ kJ}/\text{kmol}$, $B \longrightarrow C$ 的活化能 $E_2 = 1.315 \times 10^5 \text{ kJ}/\text{kmol}$, $A \longrightarrow C$ 的活化能 $E_3 = 1.197 \times 10^5 \text{ kJ}/\text{kmol}$, 理想气体常数 $R = 8.314 \text{ kJ}/(\text{kmol} \cdot \text{K})$ 。

数学模型

物料平衡方程 (程态):

$$u_s \frac{dC_A}{dz} = \rho_B r_A \quad (1)$$

$$u_s \frac{dC_B}{dz} = \rho_B r_B \quad (2)$$

$$u_s \frac{dC_C}{dz} = \rho_B r_C \quad (3)$$

热量平衡方程 (程态): $u_s \rho_g c_p \frac{dT}{dz} = \rho_B (\Delta H_1 k_1 + \Delta H_3 k_3) y_A y_{O_2} - \frac{4U}{d_t} (T - T_j) \quad (4)$

反应动力学方程 由于氧的大量过剩, 速度方程可看作拟一级。因此, 在大气压下,

$$r_A = -(k_1 + k_3) y_A y_{O_2}, r_B = k_1 y_A y_{O_2} - k_2 y_B y_{O_2}, r_C = k_2 y_B y_{O_2} + k_3 y_A y_{O_2} \quad (5)$$

其中, 反应速度常数为:

$$\ln k_1 = -\frac{113040}{RT} + 19.837, \ln k_2 = -\frac{131500}{RT} + 20.86, \ln k_3 = -\frac{119700}{RT} + 18.97 \quad (6)$$

其他公式

反应器的横截面积为 $A_c = \frac{\pi d_t^2}{4} \quad (7)$

由于总摩尔流量不变 (大多数为空气), 因此有:

$$F_t = \frac{G}{M_m} A_c \quad (8)$$

$$C_t = \frac{F_t}{A_c u_s} \quad (9)$$

转化率: $x_A = \frac{C_{A0} - C_A}{C_{A0}}, x_B = \frac{C_B}{C_{A0} - C_A}, x_C = \frac{C_C}{C_{A0} - C_A} \quad (10)$

进料 (初始) 摩尔流率: $F_{A0} = y_{A0} F_t, F_{B0} = F_{C0} = 0 \quad (11)$

程序说明 用 ode45() 求解 ODE 方程组 (1) ~ (4), 该方程组由 Equations () 定义。程序稍加修改即可考察进料条件和夹套冷却温度等重要操作参数对于轴向温度分布和转化率分布的影响 (请读者自行练习)。

程序清单 见光盘中的 *NonIsothermTR.m*。

计算结果 轴向温度分布、转化率分布和浓度分布分别示于图 4-10、图 4-11 和图 4-12 中。

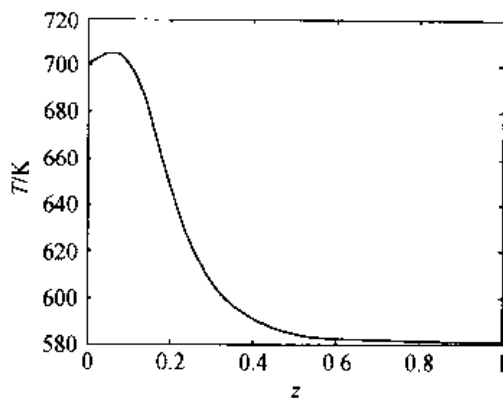


图 4-10 轴向温度分布

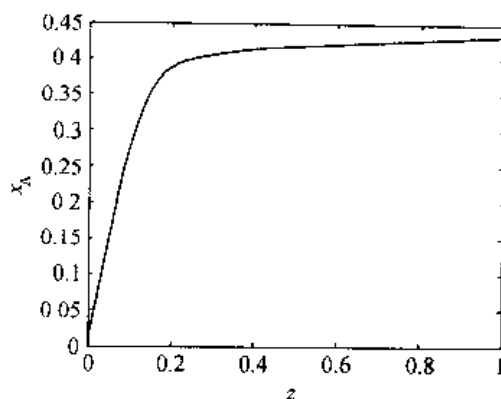


图 4-11 转化率分布

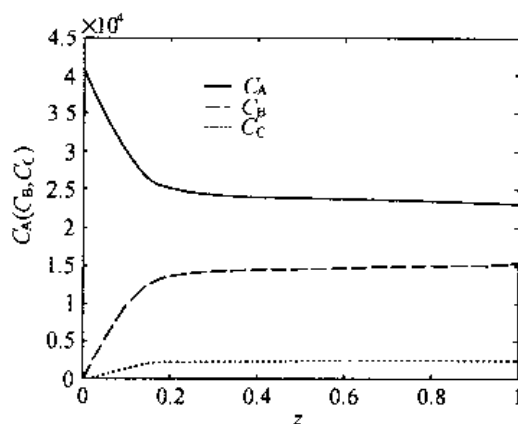


图 4-12 浓度分布

符号说明

A_c	横截面积, m^2	T	温度, K
C	浓度, $kmol/m^3$	U	传热系数, $kJ/(m^2 \cdot h \cdot K)$
c_p	比热容, $kJ/(kg \cdot K)$	u_s	线速度, m/h
d_t	管径, m	V	体积, m^3
E	活化能, $kJ/kmol$	x	转化率
G	表观质量流速, $kg/(m^2 \cdot h)$	y	摩尔分数
k	反应速率常数, $kmol/(kg \cdot h)$	z	沿管长的距离, m
M_m	平均摩尔质量, $kg/kmol$	ΔH_1	$A \longrightarrow B$ 反应热, $kJ/kmol$
F	摩尔流量, $kmol/h$	ΔH_3	$A \longrightarrow C$ 反应热, $kJ/kmol$
R	气体常数, $kJ/(kmol \cdot K)$	ρ_B	催化剂床层密度, kg/m^3
r	反应速率, $kmol/(kg \cdot h)$	ρ_G	气体混合物的密度, kg/m^3
下标			
A、B、C、D、O ₂	表示组分	0	表示入口
J	表示夹套	1、2、3	表示反应式
t	表示总量		

4.4 半连续反应器

【例 4-6】半连续反应器——苯的氯化^[5]

苯的氯化过程设备如图 4-13 所示, 在催化剂 FeCl_3 的作用下进行下列连串反应产生氯苯、二氯苯和三氯苯:

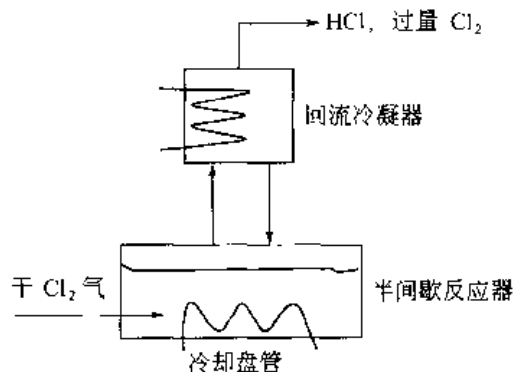
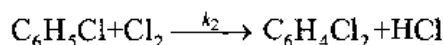
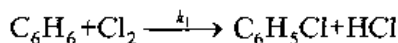


图 4-13 苯的氯化过程

图中的回流冷凝器使蒸发的氯苯、二氯苯和三氯苯回流至反应器中, HCl 和过量的 Cl_2 则从回流冷凝器顶部排放。假设:

(1) 回流冷凝器中没有液体或蒸汽滞留量, 即不涉及动态行为;

(2) 系统在等温等压下操作;

(3) 反应物中的体积变化忽略不计;

(4) Cl_2 在气液两相之间的传质阻力忽略不

计, 即 Cl_2 迅速溶解到溶液中直至溶解限度为止。已知反应器中苯的初始摩尔量为 $n_{B0} = 50(\text{kmol})$, 液体体积 $V = 1.46n_{B0}(\text{m}^3)$, 干氯气的进料量为 $F = 1.4n_{B0}(\text{kmol/h})$, 1 kmol 的初始苯最高溶解 0.12kg 的 Cl_2 , 在操作温度 55°C 下的反应速率常数 $k_1 = 510 \text{ m}^3/(\text{kmol} \cdot \text{h})$, $k_2 = 64 \text{ m}^3/(\text{kmol} \cdot \text{h})$, $k_3 = 2.1 \text{ m}^3/(\text{kmol} \cdot \text{h})$, 各反应均为二级。试计算:

- ① 氯苯达到最大产量所对应的最优时间;
- ② 二氯苯达到最大产量所对应的最优时间;
- ③ 三氯苯达到最大产量所对应的的时间。

数学模型 分别对苯、氯苯、二氯苯、三氯苯和氯气进行物料衡算, 得

$$\frac{dn_B}{dt} = -\frac{k_1 n_B n_C}{V} \quad (\text{a})$$

$$\frac{dn_M}{dt} = \frac{k_1 n_B n_C}{V} - \frac{k_2 n_M n_C}{V} \quad (\text{b})$$

$$\frac{dn_D}{dt} = \frac{k_2 n_M n_C}{V} - \frac{k_3 n_D n_C}{V} \quad (\text{c})$$

$$\frac{dn_I}{dt} = -\frac{k_3 n_D n_C}{V} \quad (\text{d})$$

$$\frac{dn_C}{dt} = F - \frac{k_1 n_B n_C}{V} - \frac{k_2 n_M n_C}{V} - \frac{k_3 n_D n_C}{V} \quad (\text{e})$$

$$\text{最大的氯浓度为} \quad n_{C, \max} = 0.12n_{B0} \quad (\text{f})$$

程序说明 用 `ode45()` 求解由 `MassBalances()` 定义的 ODE 方程组 (a)~(e)。由于每 1 kmol 的初始苯最高溶解 0.12kg 的 Cl_2 , 因此 `MassBalances()` 中加入判断语句, 即如果 $n_C > 0.12 \cdot n_{B0}$, 则取 $n_C = 0.12 \cdot n_{B0}$, $dn_C/dt = 0$ 。

程序清单 见光盘中的 `SemiBatchReactor.m`。

计算结果 各组分浓度分布于图 4-14 中。由图可见, 三氯苯达到最大产量所对应的的时间为 ∞ 。氯苯和二氯苯达到最大产量所对应的最优时间分别为 0.76 h 和 1.71 h。

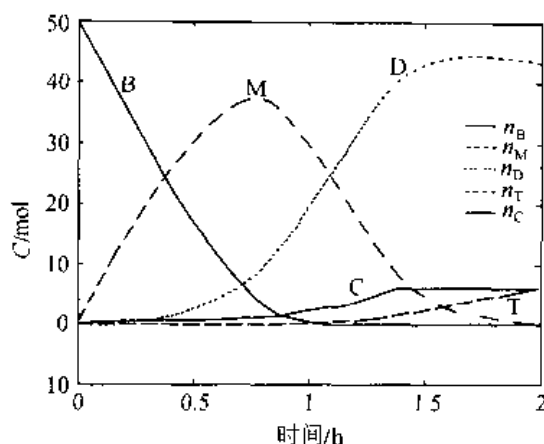


图 4-14 浓度分布

符号说明

F	干氯气的进料量	kmol/h
n_i	组分 i 的物质的量 (摩尔数)	mol
k_1, k_2, k_3	三个反应式的反应速率常数	$\text{m}^3/(\text{kmol} \cdot \text{h})$
t	反应时间	h
V	反应器中的液体体积	m^3
下标		
B、M、D、T、C	分别表示苯、氯苯、二氯苯、三氯苯和氯气	
0	初始	

4.5 传质过程

【例 4-7】连续多组分精馏^[1]

在一精馏塔中进行多组分液体混合物的分离, 该混合物含有苯、苯乙烯和甲苯三个组分。已知进料流量 $F = 40 \text{ kmol/hr}$, 回流比 $R = 5$, 进料组成 (摩尔分数) $x_1 = 0.6$, $x_2 = 0.25$, 塔顶冷凝器中的滞液量 $M_1 = 75 \text{ kmol}$, 塔板滞液量 $M = 10 \text{ kmol}$, 塔釜中的滞液量 $M_N = 150 \text{ kmol}$, 进料状态 $q = 1$ (饱和), 相对挥发度: $\alpha_1 = 2.75$, $\alpha_2 = 1$, $\alpha_3 = 0.4$, 从塔釜蒸发上来的蒸汽流量 $V' = 150 \text{ kmol/h}$, 塔板总数 $N_t = 10$ (包括塔顶冷凝器和塔釜), 进料板位置 $N_f = 5$ 。

- ① 求塔顶和塔釜产品从进料开始直至稳态的动态浓度曲线;
- ② 画出稳态时精馏塔各塔板上的浓度曲线;
- ③ 研究操作变量 (进料流量、进料组成和回流比) 的变化对精馏的影响;

数学模型

对塔顶冷凝器 (reflux drum) 的任意组分 j (这里 $j = 1, 2, \dots, n$),

$$M_1 \frac{dx_{1,j}}{dt} = Vy_{2,j} - (L + D)x_{1,j} \quad i = 1 \quad (1)$$

对精馏段 (column enriching section) 第 i 块板的任意组分 j ,

$$M \frac{dx_{i,j}}{dt} = L(x_{i-1,j} - x_{i,j}) + V(y_{i-1,j} - y_{i,j}) \quad i = 2, 3, \dots, N_f - 1 \quad (2)$$

对进料板 (column feedplate) 的任意组分 j ,

$$M \frac{dx_{N_f,j}}{dt} = Lx_{N_f-1,j} - L'x_{N_f,j} + V'y_{N_f+1,j} - Vy_{N_f,j} + Fz_{i,j} \quad i = N_f \quad (3)$$

对提馏段 (column stripping section) 第 i 块板的任意组分 j ,

$$M \frac{dx_{i,j}}{dt} = L'(x_{i-1,j} - x_{i,j}) + V'(y_{i+1,j} - y_{i,j}) \quad i = N_f+1, N_f+2, \dots, N_t-1 \quad (4)$$

对塔釜 (column reboiler) 的任意组分 j ,

$$M_B \frac{dx_{B,j}}{dt} = L'x_{N,j} - Wx_{B,j} - V'y_{B,j} \quad i = N_t \quad (5)$$

在精馏段中

$$L = RD \quad (6)$$

$$V = (R+1)D \quad (7)$$

在提馏段中

$$L' = L + qF \quad (8)$$

$$V = V' - (1-q)F \quad (9)$$

若饱和液体进料 (泡点进料), 则 $q=1$ 。

气液相平衡关系

$$y_{i,j} = \frac{\alpha_{i,j} x_{i,j}}{\sum_{j=1}^n \alpha_{i,j} x_{i,j}} \quad (10)$$

式中, i 为塔板序号 ($i=2, \dots, N_t-1$), j 为组分序号 ($j=1, 2, \dots, n$)。

程序说明 用 ode45() 求解由 DistMassBalances() 定义的物料平衡方程组 (1) ~ (5)。由于是三元组分, 各板上都满足 $x_3=1-x_1-x_2$ (其中 x_1 、 x_2 和 x_3 均为向量, 分别表示组分 1、2 和 3 在各个塔板上的液相摩尔分数), 故只需对组分 1 和 2 的有关动态方程进行求解。[x_1 x_2] 初值的选取依据: 开车时塔内所有板上的 x_1 和 x_2 分别与进料的 z_1 和 z_2 相同, 故初值 [x_1 x_2] 中向量 x_1 的各个元素都取为 z_1 , 向量 x_2 的各个元素都取为 z_2 。

程序清单 ConDistill.m

```
function ConDistill % 连续多组分 (三元) 精馏塔的模拟计算
clear all; clc
global F z1 z2 z3 R alpha1 alpha2 alpha3 M1 MB M Nt Nf V1 V D L L1 W
F = 40; R = 5; % F: 进料流量, kmol/h, R: 回流比
z1 = 0.6; z2 = 0.25; z3 = 1-z1-z2; % 苯、甲苯和苯乙烯的进料组成 (摩尔分数)
M1 = 75; M = 10; MB = 150; % 塔顶冷凝器、塔板和塔釜中的滞液量(kmol)
q = 1; tf = 50; dt = 1; % 饱和进料
alpha1 = 2.75; alpha2 = 1; alpha3 = 0.4; % 相对挥发度
Nt = 10; Nf = 5; % Nt: 塔板总数, Nf: 进料位置
V1 = 150; % 从塔釜蒸发上来的蒸汽流量(kmol/hr)
V = V1-(1-q)*F; D = V/(R+1); L = V-D; % 精馏段
L1 = L+F; W = L1-V1; % 提馏段

% 初始化 x1 和 x2---开车时塔内所有板上的 x1 和 x2 分别与进料的 z1 和 z2 相同
x1 = z1 * ones(1,Nt); x2 = z2 * ones(1, Nt);

[t, y] = ode45(@DistMassBalances, [0:dt:tf], [x1 x2])

% 输出结果
x1 = y(:, 1:Nt); x2 = y(:, Nt+1:2*Nt); x3 = 1-x1-x2; % 苯、甲苯和苯乙烯的液相摩尔分数
```

```

plot(t, x1(:,1), 'r-', t, x2(:,1), 'k--', t, x3(:,1), 'b:', t, x1(:,end), 'r-', t, x2(:,end), 'k-', t, x3(:,end), 'b.--')
xlabel('Time (h)'), ylabel('x_1_1, x_1_2, x_1_3, x_N_1, x_N_2, x_N_3')
title('塔顶和塔釜产品从进料开始直至稳态的动态浓度曲线')
legend('x_1_1', 'x_1_2', 'x_1_3', 'x_N_1', 'x_N_2', 'x_N_3')
% 稳态图
figure, plate = 1:Nt; plot(plate, x1(end,:), 'r-', plate, x2(end,:), 'k-', plate, x3(end,:), 'b:')
xlabel('塔板'), ylabel('稳态时苯, 甲苯, 苯乙烯的组成'), title('稳态时精馏塔的浓度曲线')
legend('苯', '甲苯', '苯乙烯')

% -----
function dydt = DistMassBalances(t, y) % 物料平衡方程组
global F z1 z2 z3 R alpha1 alpha2 alpha3 M1 MB M Nt Nf V1 V D L L1 W
x1 = y(1:Nt); x2 = y(Nt+1:2*Nt); x3 = 1-x1-x2; % 苯、甲苯和苯乙烯的液相摩尔分数

% 气相平衡
denom = alpha1*x1+alpha2*x2+alpha3*x3; y1 = alpha1*x1./denom; y2 = alpha2*x2./denom;

% 对塔顶冷凝器 (i = 1)
i = 1; dx1dt(i) = (V*y1(i+1)-(L+D)*x1(i))/M1; dx2dt(i) = (V*y2(i+1)-(L+D)*x2(i))/M1;

% 精馏段 (i = 2~Nf-1)
for i=2:Nf-1
    dx1dt(i) = (L*(x1(i-1)-x1(i))+V*(y1(i+1)-y1(i)))/M;
    dx2dt(i) = (L*(x2(i-1)-x2(i))+V*(y2(i+1)-y2(i)))/M;
end

% 进料板 (i = Nf)
i = Nf;
dx1dt(i) = (F*z1+L*x1(i-1)-L1*x1(i)+V1*y1(i+1)-V*y1(i))/M;
dx2dt(i) = (F*z2+L*x2(i-1)-L1*x2(i)+V1*y2(i+1)-V*y2(i))/M;

% 提馏段 (Nf+1~Nt-1)
for i=Nf+1:Nt-1
    dx1dt(i) = (L1*(x1(i-1)-x1(i))+V1*(y1(i+1)-y1(i)))/M;
    dx2dt(i) = (L1*(x2(i-1)-x2(i))+V1*(y2(i+1)-y2(i)))/M;
end

% 塔釜 (i = Nt)
i = Nt;
dx1dt(i) = (L1*x1(i-1)-V1*y1(i)-W*x1(i))/MB;
dx2dt(i) = (L1*x2(i-1)-V1*y2(i)-W*x2(i))/MB;

dydt = [dx1dt dx2dt]';

```

计算结果 (1) 塔顶和塔釜产品从进料开始直至稳态的动态浓度曲线如图 4-15 所示(经

过 30 h 基本达到稳态); (2) 稳态时各塔板上的浓度曲线示于图 4-16 中; (3) 读者对程序稍微修改即可模拟操作变量 (进料流量、进料组成和回流比) 的变化对精馏的影响, 这里从略。

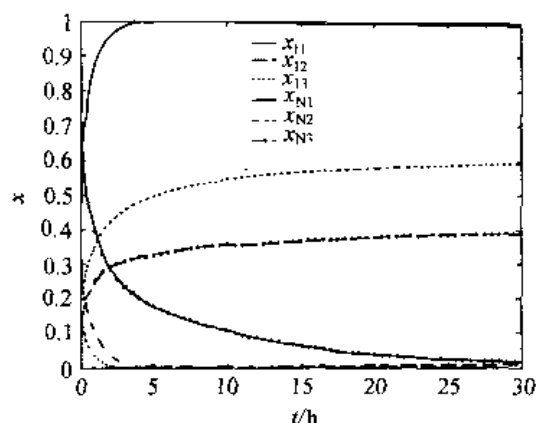


图 4-15 塔顶和塔釜产品的动态浓度曲线

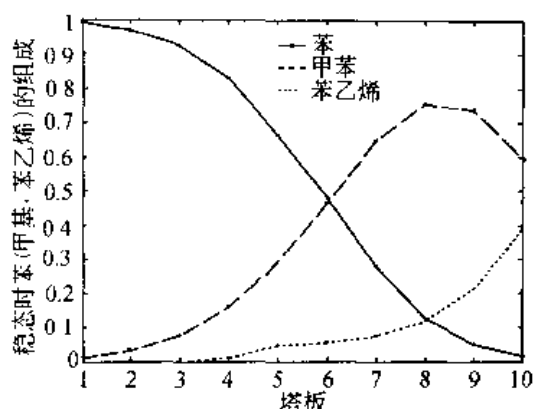


图 4-16 稳态时精馏塔各塔板上的浓度曲线

符号说明

D	塔顶馏出液流量, kmol/h	R	回流比
F	进料流量, kmol/h	V	蒸汽流量, kmol/h
L	液体流量, kmol/h	x	液相摩尔分数
M	液体滞流量, kmol/h	y	气相摩尔分数
N	塔板数	z	进料组成(摩尔分数)
Q	进料状态	α	相对挥发度
下标			
'	提馏段	i	组分 i
l	塔顶冷凝器	j	组分数
B	再沸器	N	塔板总数
f	进料板	t	总数 (包括塔顶冷凝器和塔釜)

【例 4-8】 气体吸收塔模拟计算^[6]

一个气体吸收塔有六块理论板, 每块板有 5kg 滞留液体。在连续操作周期之间, 惰性气体穿过吸收塔, 纯溶剂以 100kg/h 的递减流量加入。当再启动此过程时, 含可溶组分 5% (质量) 的气体以 500kg/h 的流量加入吸收塔的底部。如果吸收系数是 $y = 0.40x$ ($k = 0.40$), 为保持出口气体浓度低于 0.2%, 问经多长时间才需增加液体流量?

数学模型

如图 4-17 所示, 作第 i 块板的物料衡算,

$$Gy_{i-1} + Lx_{i+1} - Gy_i - Lx_i = H \frac{dx_i}{dt} \quad (1)$$

$$\text{已知} \quad y_i = kx_i \quad (2)$$

故方程 (1) 可消去 x_i 和 x_{i+1} , 于是 $\alpha y_{i-1} + y_{i+1} - \alpha y_i - y_i = \frac{H}{L} \frac{dy_i}{dt}$

$$\text{即} \quad \frac{dy_i}{dt} = [\alpha y_{i-1} - (\alpha + 1)y_i + y_{i+1}] \frac{L}{H} \quad (i = 1, 2, \dots, 6) \quad (3)$$

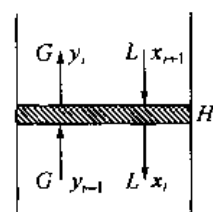


图 4-17 通过气体吸收塔第 i 块板的流动

$$\text{式中} \quad \alpha = k \frac{G}{L} \quad (4)$$

y_0 、 y_7 分别是进入塔底和离开塔顶的气体浓度。

显然, (3) 是由 6 个方程组成的 ODE 方程组, 其初始条件为

$$t = 0, y_i = 0 \quad (i = 1, 2, \dots, 6) \quad (5)$$

已知条件: $H = 5 \text{ kg}$, $L = 100 \text{ kg/h}$, $y_0 = 0.05$, $y_7 = 0$, $G = 500 \text{ kg/h}$, $k = 0.40$ 。

程序说明

方程组 (3) 可用有限差分法求解, 即取步长 Δt , 用有限差分代替方程 (3) 中的导数项 (沿时间离散), 得到代数方程组, 然后沿时间增量进行迭代计算, 每步迭代求解一次代数方程组。这里直接使用 `ode45()` 进行求解, 以避免离散之麻烦。

程序中, `MassBalanceEqs()` 按方程 (3) 定义常微分方程组, 先用 `ode45()` 积分求得浓度的动态数据, 再用样条插值函数 `spline()` 计算出口气体浓度 y_6 为 0.2% 时所需的时间。

需注意的是, 由于料液成分的阶跃变化, 在 $t=0$ 时, 初始条件 (5) 的 $y_0 = 0$ 与已知条件 $y_0 = 0.05$ 有矛盾。经验表明, 取 y_0 等于两个矛盾条件的算术平均 (0.025) 较合理。所以函数 `MassBalanceEqs()` 中增加条件语句: 当 $t=0$ 时, $y_0=C0$; 当 $t>0$ 时, $y_0=C1$ (这里 $C0=0.025$, $C1=0.05$)。实际上, 本例若不考虑 y_0 的阶跃变化 (不取两个矛盾条件的算术平均), 而取 $C0=0.05$ 时, 计算结果基本不变 (读者不妨试一试)。

也可以不用 `spline()` 等插值方法, 而同时使用 `fzero()` 和 `ode45()` 求解出口气体浓度 y_6 为 0.2% 时所需的时间, 程序结构类似例 4-13 的程序 `PFRlength.m`, 后者同时使用 `fzero()` 和 `quadl()` 求解。但这种方法反复迭代, 速度要慢一些。

程序清单 *Absorption.m*

```
function Absorption          % 气体吸收模拟计算
clear all; clc
global alpha C0 C1 y7 L H
H = 5; L = 100; C0 = 0.025; C1 = 0.05; y7 = 0;      % H: liquid holdup(kg), L: kg/h
G = 500; k = 0.4; alpha = k*G/L;                  % G: kg/h
dt = 1/120; stoptime = 11*dt;
[t, y] = ode45(@MassBalanceEqs, [0:dt:stoptime], [0 0 0 0 0 0]);
tr = spline(y(:,6), t, 0.002);                     % tr: the time required, h

% Output results
disp('Results (t — hour, yi — wt%):')
disp('      t      y1      y2      y3      y4      y5      y6')
disp([t y*100]), fprintf('\nTime required: %.1f (min)', tr*60)

% -----
function dydt = MassBalanceEqs(t, y) % 吸收塔第 i (i=1~6) 块塔板的质量平衡
global alpha C0 C1 y7 L H
if t==0, y0 = C0; end
if t>0, y0 = C1; end
```

```

dydt(1) = (alpha*y0-(alpha+1)*y(1)+y(2))*L/H;           % i = 1
for i=2:5, dydt(i) = (alpha*y(i-1)-(alpha+1)*y(i)+y(i+1))*L/H; end % i = 2...5
dydt(6) = (alpha*y(5)-(alpha+1)*y(6)+y7)*L/H;           % i = 6
dydt = dydt';

```

计算结果 浓度的动态数据如表 4-1 所示。出口气体浓度 y_6 为 0.2% 时所需的时间为 4.8 min, 即为保持出口气体浓度小于 0.2%, 在浓的气体进入吸收塔后 4.8min 必须增加液体流量。

表 4-1 吸收塔内各级气体的组分 (质量分数)

t/h	y1	y2	y3	y4	y5	y6
0	0	0	0	0	0	0
0.0083	1.3211	0.2018	0.0214	0.0017	0.0001	0.0000
0.0167	2.1670	0.6058	0.1225	0.0192	0.0025	0.0003
0.0250	2.7369	1.0495	0.3019	0.0688	0.0129	0.0020
0.0333	3.1395	1.4701	0.5338	0.1563	0.0381	0.0078
0.0417	3.4359	1.8474	0.7929	0.2792	0.0827	0.0206
0.0500	3.6618	2.1781	1.0606	0.4306	0.1486	0.0431
0.0583	3.8391	2.4657	1.3249	0.6025	0.2351	0.0770
0.0667	3.9816	2.7154	1.5786	0.7874	0.3399	0.1229
0.0750	4.0984	2.9327	1.8181	0.9790	0.4599	0.1804
0.0833	4.1958	3.1225	2.0417	1.1724	0.5916	0.2486
0.0917	4.2780	3.2892	2.2489	1.3639	0.7316	0.3257

符号说明

G	气体流量	kg/h	t	时间	h
H	液体滞留量	kg	x	液相中可溶组分的浓度	
K	吸收系数		y	气相中可溶组分的浓度	
L	纯溶剂流量	kg/h	α	中间变量	$\alpha = kG/L$

下标

i 第 i 块塔板

【例 4-9】多组分气体扩散^{[7][11]}

在 55℃、20.265kPa(0.2 atm)下, 气体 A 和 B 扩散通过静止的气体 C。该过程涉及到两个位置点之间 (间离为 10^{-3} m) 的分子扩散, 各点处的组成如表 4-2 所示。

- (1) 使用 Stefan-Maxwell 方程计算组分 A 和 B 从位置 1 到位置 2 的摩尔扩散通量;
- (2) 绘制气体摩尔分数 x 随位置 z 的变化。

表 4-2 多组分扩散

组分	位置点 1 处的浓度/(kmol/m ³)	位置点 2 处的浓度/(kmol/m ³)	0.2 atm 下的扩散系数/(m ² /s)
A	2.229×10^{-4}	0	$D_{AC} = 1.075 \times 10^{-4}$
B	0	2.701×10^{-3}	$D_{BC} = 1.245 \times 10^{-4}$
C	7.208×10^{-3}	4.730×10^{-3}	$D_{AB} = 1.470 \times 10^{-4}$

数学模型

z 方向的 Stefan-Maxwell 方程为:

$$\frac{dC_i}{dz} = \sum_j^n \frac{(x_i N_j - x_j N_i)}{D_{ij}} \quad (1)$$

式中, C_i 为组分 i 的扩散浓度, kmol/m^3 ; x_i 为组分 i 的摩尔分数; N_i 为组分 i 的摩尔通量, m^2/s ; n 为组分数; D_{ij} 为组分 i 和 j 的双组分扩散系数, m^2/s 。
应用式 (1) 于三组分混合物, 得

$$\frac{dC_A}{dz} = \frac{x_A N_B - x_B N_A}{D_{AB}} + \frac{x_A N_C - x_C N_A}{D_{AC}} \quad (2)$$

$$\frac{dC_B}{dz} = \frac{x_B N_A - x_A N_B}{D_{AB}} + \frac{x_B N_C - x_C N_B}{D_{BC}} \quad (3)$$

$$\frac{dC_C}{dz} = \frac{x_C N_A - x_A N_C}{D_{AC}} + \frac{x_C N_B - x_B N_C}{D_{BC}} \quad (4)$$

其中考虑了 $D_{BA} = D_{AB}$ 、 $D_{CA} = D_{AC}$ 、 $D_{CB} = D_{BC}$ 。

由于组分 C 是静止的, 因此有 $N_C = 0$ (5)

所以, 该问题需要优化 N_A 和 N_B , 直至满足表 4-2 中的边界条件。

程序说明

定义误差函数 $\varepsilon(C_A) = 0 - C_A|_{z=0.001}$ (6)

$$\varepsilon(C_B) = 2.701 \times 10^{-3} - C_B|_{z=0.001} \quad (7)$$

收敛时, 误差函数趋向于 0。

简单的优化方法是先固定 N_B , 通过使 $\min_{N_A} |\varepsilon(C_A)| = 0$ 找到最优 N_A , 再固定最优值 N_A , 搜索 N_B , 使 $\min_{N_B} |\varepsilon(C_B)|$ 目标函数最小化。这里使用多变量优化方法, 同时搜索 N_A 和 N_B , 优化目标函数定义为:

$$\min_{N_A, N_B} |\varepsilon(C_A)| + |\varepsilon(C_B)| \quad (8)$$

程序中, 目标函数 (8) 式由函数 `ObjFunc()` 定义, 且 `ObjFunc()` 内部调用 `ode45()` 求解由 `ODEs()` 定义的方程组 (2)、(3) 和 (4)。采用 Nelder-Mead 优化算法函数 `fminsearch()`, 对 `ObjFunc()` 定义的目标函数进行优化, 求得最优的 N_A 和 N_B 。

程序清单 *MultiCompDiff.m*

```
function MultiCompDiff
clear all; clc
global CT NA NB NC DAB DBC DAC
NA0 = 2.396e-5; NB0 = -3.363e-4; NC = 0;
DAB = 1.47e-4; DBC = 1.245e-4; DAC = 1.075e-4; CT = 0.2/(82.057e-3*328);
z0 = 0; zf = 0.001; CA0 = 2.229e-4; CB0 = 0; CC0 = 7.208e-3; CAf = 0;
CBf = 2.701e-3; CCf = 4.730e-3; zspan = [z0 zf]; y0 = [CA0 CB0 CC0];

flux = fminsearch(@ObjFunc, [NA0 NB0], [], zspan, y0, CAf, CBf) % 优化
NA = flux(1); NB = flux(2);

[z, y] = ode45(@ODEs, zspan, y0, [], NA, NB); % 解 ODE 方程组
xA = y(:,1)/CT; xB = y(:,2)/CT; xC = y(:,3)/CT;
% 输出结果
```



```

fprintf('组分 A 和 B 从位置 1 到位置 2 的摩尔扩散通量分别是: NA = %.2e, NB = %.2e',NA,NB)
plot(z, xA, '-', z, xB, '-.', z, xC, '--')
xlabel('Distance (z), m'), ylabel('Mole fraction'), legend('x_A', 'x_B', 'x_C')

% -----
function f = ObjFunc(flux, zspan, y0, CAf, CBf)
NA = flux(1); NB = flux(2);
[z, y] = ode45(@ODEs, zspan, y0, [], NA, NB);
f = abs(CAf - y(end,1)) + abs(CBf - y(end,2))

% -----
function dydz = ODEs(z, y, NA, NB)
global CT NC DAB DBC DAC
CA = y(1); CB = y(2); CC = y(3); xA = CA/CT; xB = CB/CT; xC = CC/CT;
dCA dz = (xA*NB-xB*NA)/DAB+(xA*NC-xC*NA)/DAC;
dCB dz = (xB*NA-xA*NB)/DAB+(xB*NC-xC*NB)/DBC;
dCC dz = (xC*NA-xA*NC)/DAC+(xC*NB-xB*NC)/DBC;
dydz = [dCA dz; dCB dz; dCC dz];

```

计算结果 组分 A 和 B 从位置 1 到位置 2 的摩尔扩散通量分别是: $N_A = 1.80\text{e-}5$, $N_B = -4.08\text{e-}4$ 。摩尔分数 x 随位置 z 的变化图略。

【例 4-10】 预测口服药在人体中的吸收^[20]

预测口服药在人体中的吸收是药物研究很感兴趣的一项内容，它可以帮助加速新药的开发。下面通过建模预测高度可溶的药被吸收的药物分率。

数学模型

假设人体小肠是一个半径为 R 、长为 L 的圆管，初始浓度为 C_0 的药物以体积流率 Q 进入此圆管。假设稳态，考虑从圆管中的药物吸收，则分别对药物颗粒和溶液相进行微观质量衡算，可导出如下的 ODE 方程组：

$$\frac{dr^*}{dz^*} = -\frac{Dn}{3} \left(\frac{1-C^*}{r^*} \right) \quad (1)$$

$$\frac{dC^*}{dz^*} = Dn Do r^* (1-C^*) - 2 An C^* \quad (2)$$

其中
$$z^* = \frac{z}{L}, \quad r^* = \frac{r_p}{r}, \quad C^* = \frac{C}{C_0} \quad (3)$$

$$Do = \frac{C_0}{C_s} = \frac{M_0/V_0}{C_s}, \quad An = \frac{\pi R L P_{\text{eff}}}{Q}, \quad Dn = \frac{\pi R^2 L}{Q} \left(\frac{3DC_0}{\rho r_0^2} \right) \quad (4)$$

式中， r_p 为药物颗粒半径； M_0 为药剂量； V_0 为服药时与药同时喝下去的水体积； C_s 为生理溶解度； P_{eff} 为有效渗透系数； D 为扩散系数， ρ 为颗粒密度； r_0 为药物颗粒的初始半径。 Do 、 An 和 Dn 分别称为药剂数 (dose number)、吸收数 (absorption number) 和量纲为一的溶解数 (dissolution number)。

方程 (1)、(2) 的初始条件为： $z^* = 0, r^*(0) = 1, C^*(0) = 1 \quad (5)$

$$\text{被吸收的药物分率可由下式估算: } F_a = 1 - (r^*)_{z=1}^3 - \left(\frac{C^*}{Do} \right) \bigg|_{z=1} \quad (6)$$

根据地高辛 (digoxin) — 强心剂药物颗粒的物理化学/生理参数 (密度、生理溶解度、剂量、初始半径、扩散系数等) 已得到 $Do = 0.08$ 、 $An = 10.0$ 和 $Dn = 0.6$, 试求解由式 (1)、式 (2)、式 (5) 组成的 ODE-IVP 问题, 绘制此药物颗粒的半径变化曲线及其浓度分布, 并计算被吸收的药物分率。

程序清单 见光盘中的 *DrugAbsorp.m*, 程序中的 z 表示 z^* 。

计算结果 被吸收的药物分率为 $F_a = 0.4869$ 。药物颗粒半径变化曲线及药物浓度分布图从略。

4.6 伴有反应的扩散过程

反应-扩散方程通常是一个二阶非线性微分方程的 BVP 问题。在反应工程教科书和参考书中, 一般都给出等温催化剂 (片状、圆柱形和球形) 一级反应内扩散有效因子的分析解, 对于等温催化剂非一级反应或者非等温催化剂, 内扩散有效因子通常只能通过数值求解。

【例 4-11】 计算片状催化剂中等温一级不可逆反应的有效因子 (等温内部效率因子) ^[8]。

已知 $f(C) = C$, Thiele 模数 $\Phi = 6$ 。

$$\text{数学模型} \quad \frac{d^2 C}{dx^2} = \phi^2 f(C), \quad 0 < x < 1 \quad (a)$$

$$\text{B.C.} \quad \frac{dC}{dx}(0) = 0, \quad C(1) = 1 \quad (b)$$

式中, C 为浓度; x 为催化剂的厚度; Φ 为 Thiele 模数。

对于片状催化剂, 等温有效因子为:

$$\eta = \frac{\int_0^1 f(C) dx}{\int_0^1 f(C_s) dx} \quad (c)$$

由于 $f(C) = C$, $C_s = C(1) = 1$, 故 (c) 式变为:

$$\eta = \int_0^1 C dx \quad (d)$$

程序说明

方程 (a)、(b) 是二阶 ODE 方程, 通常先降阶为一阶 ODE 方程组, 然后再求解。

令 $y_1 = C$, $y_2 = dC/dx$, 则方程 (a)、(b) 转变为一阶 ODE 方程组:

$$\begin{cases} y_1' = y_2 \\ y_2' = \phi^2 f(y_1) \end{cases} \quad (e)$$

$$\text{B.C.} \quad y_1(1) = 1, \quad y_2(0) = 0 \quad (f)$$

这是 ODE-BVP 问题, 其中一个微分方程的边界条件是初始条件 ($x = 0$), 另一个微分方程的边界条件是最终条件 ($x = 1$), 所以称为 split boundary conditions。对此问题, 可先用 *bvpinit()* 和 *bvp4c()* 求出沿催化剂厚度方向的浓度分布, 即离散数据 $y_1(x)$ 。再利用 *spline()* 由 $y_1(x)$ 构造样条函数, 该函数亦即式 (d) 的被积函数, 然后用 *quadl()* 根据式 (d) 积分求出 η 。

程序中, *ODEs()* 定义方程组 (e), *BCfun()* 则定义其边界条件 (f) 式。而 *IntFunc()* 定义式 (d) 的被积函数, 即利用 *spline()* 由离散数据 $y_1(x)$ 构造样条函数。

程序清单 *PorousPlateCat_1Ord.m*

```
function PorousPlateCat_1Ord % 多孔片状催化剂颗粒中的扩散和一级不可逆等温反应 clear
all; clc
global fai
fai = 6; a = 0; b = 1; % fai: Thiele 模数, 积分区间[a, b]
solinit = bvpinit(linspace(a, b, 100), [0 0]); % 以猜测值 y1(x)=0, y2(x)=0 对解进行初始化
sol = bvp4c(@ODEs, @BCfun, solinit); % 求解 ODE-BVP 问题

% 绘制浓度分布图
plot(sol.x, sol.y(1,:), 'b-')
xlabel('Dimensionless Thickness'), ylabel('Dimensionless Concentration')

eta = quadl(@IntFunc, a, b, [], [], sol.x, sol.y(1,:)) % 计算催化剂的有效因子
fprintf('\n 催化剂的有效因子为: \eta = %.4f, eta)

% -----
function dydx = ODEs(x, y)
global fai
dydx = [y(2); fai^2*y(1)];

% -----
function hc = BCfun(ya, yb)
bc = [yb(1)-1; ya(2)];

% -----
function f = IntFunc(x, xi, yi)
f = spline(xi, yi, x);
```

计算结果 催化剂的有效因子为: $\eta = 0.1667$ 。

【例 4-12】 多孔球形催化剂在等温条件下的一级不可逆反应^[9]

已知反应系统为环己烷催化脱氢, 所用的 $Pt-\gamma$ 氧化铝催化剂为 5 mm 球形颗粒, 操作温度为 700 K, 在此温度下, $k = 4 \text{ s}^{-1}$, $D = 0.05 \text{ cm}^2/\text{s}$, 试计算催化剂颗粒内环己烷的浓度分布及催化剂的有效因子。

数学模型 多孔球形颗粒催化剂内的质量传递方程为

$$\frac{d^2 C}{dr^2} + \frac{2}{r} \frac{dC}{dr} = \phi^2 \frac{f(C)}{C_s}, \quad 0 < r < 1 \quad (1)$$

$$\text{B.C.} \quad \frac{dC}{dr}(0) = 0, \quad C(1) = 1 \quad (2)$$

式中, C 为浓度; r 为催化剂颗粒的半径; C_s 为催化剂颗粒表面浓度, ϕ 为 Thiele 模数。

$$\phi = r_p \sqrt{\frac{k}{D}} \quad (3)$$

```

end
dydr = [dy1dr; dy2dr];

% -----
function bc = BCfun(ya, yb)
bc = [yb(1)-1; ya(2)];

% -----
function f = IntFunc1(r, ri, yi)
f = spline(ri, yi, r) .* r.^2;

% -----
function f = IntFunc2(r)
global Cs
f = Cs * r.^2;

```

计算结果 催化剂的有效因子为: $\eta = 0.7726$ 。

【例 4-13】 多孔球形催化剂在等温条件下的二级不可逆反应^[9]

在[例 4-13]的基础上, 若反应速率函数改为二级反应: $f(C) = C^2$, Thiele 模数 ϕ 不变, 重复计算催化剂颗粒内环己烷的浓度分布及催化剂的有效因子。

解 只需对上例程序 PorousSphereCat_1Ord.m 修改相应的反应速率函数即可, 程序见光盘中的 PorousSphereCat_2Ord.m。计算结果为: 催化剂的有效因子 $\eta = 0.6742$ 。

[例 4-11]~[例 4-13]都是先求解物料衡算微分方程 (ODE-BVPs) 得到浓度分布, 然后根据等温有效因子的定义式, 通过数值积分求得有效因子 η 。[例 4-14]和[例 4-15]将详细推导反应-扩散过程的数学模型, 并将有效因子的定义式变化为一个 ODE 方程, 然后与物料衡算方程一起构成 ODE 方程组 (ODE-BVPs 问题), 最后对此方程组进行数值求解, 得到 η 。

【例 4-14】 多孔球形催化剂在等温条件下的一级不可逆反应^[10]

数学模型

要计算催化剂的有效因子, 必须在催化剂的几何形状内取微元进行物料衡算, 并结合 Fick 扩散定律, 导出 ODE 方程。

在多孔催化剂中 A 的物料衡算

对球形催化剂, 在催化剂球形颗粒内取微元体积, 在此微元内对组分 A 进行物料衡算:

$$\frac{d(N_A r^2)}{dr} = -k'' a C_A r^2 \quad (1)$$

$$\text{B.C.} \quad (N_A r^2) \Big|_{r=0} = 0 \quad (\text{颗粒中心处的传质通量为 } 0) \quad (2)$$

式中, N_A 为反应物 A 的传质通量; r 为球形颗粒的半径; k'' 为基于颗粒体积的一级反应速率常数; a 为单位体积颗粒的表面积; C_A 为反应物的浓度。

反应物 A 的 Fick 扩散方程

$$\frac{dC_A}{dr} = \frac{N_A}{-D_e} \quad (3)$$

$$\text{B.C.} \quad C_A \Big|_{r=R} = C_{AS} \quad (\text{颗粒中心处的传质通量为 } 0) \quad (4)$$

式中, D_e 为多孔催化剂颗粒中反应物 A 的有效扩散系数; R 为球形颗粒半径。

由于方程 (1) 的因变量为复合变量 ($N_A r^2$), 故下式可为方程 (3) 提供 N_A 的计算:

$$N_A = \frac{N_A r^2}{r^2} \quad (5)$$

催化剂的有效因子 即催化剂颗粒内的平均反应速率与催化剂颗粒表面处的反应速率:

$$\eta = \frac{\text{催化剂颗粒内的平均反应速率}}{\text{催化剂颗粒表面处的反应速率}} \quad (6)$$

即

$$\eta = \frac{\int_0^R k'' a C_A (4\pi r^2) dr}{k'' a C_{AS} \left(\frac{4}{3} \pi R^3 \right)} = \frac{3}{C_{AS} R^3} \int_0^R C_A r^2 dr \quad (7)$$

$$\text{对 } r \text{ 求导, 得} \quad \frac{d\eta}{dr} = \frac{3C_A r^2}{C_{AS} R^3} \quad (8)$$

$$\text{I.C.} \quad \eta|_{r=0} = 0 \quad (9)$$

而有效因子是 $r=R$ 处的 η 值。

所以, 由式 (1)、式 (3) 和式 (8) 可构成 ODE 方程组, 其边界条件为式 (2)、式 (4) 和式 (9)。

程序说明 这是 ODE-BVPs 问题, C_A 初始值未知, 可用打靶法 (Shooting technique) 确定 C_A 的初始条件, 过去的文献多介绍这种方法。本程序直接使用 MATLAB 的配置法函数 `bvp4c()` 进行求解, 其中函数 `ODEs()` 定义方程式 (1)、式 (3) 和式 (8), 并用 `if...else...end` 语句避免方程 (5) 中除数为 0 的情况出现。函数 `BCfun()` 则定义其边界条件式 (2)、式 (4) 和式 (9)。

程序清单 *DiffReact_SphereCat.m*

```
function DiffReact_SphereCat
clear all; clc
global CAs R De kppa
CAs = 0.2; R = 0.5; De = 0.1; kppa = 6.4; a = 0; b = R; % 积分区间[a, b]
solinit = bvpinit(linspace(a, b, 50), [0 CAs 0]); % 对解进行初始化
sol = bvp4c(@ODEs, @BCfun, solinit); % 求解 ODE-BVP 问题
y = deval(sol, [a b]); fprintf('催化剂的有效因子为 \eta = %.3f, y(end,2))

% -----
function dydr = ODEs(r, y)
global CAs R De kppa
NAr = y(1); CA = y(2); eta = y(3);
dNArdr = -kppa*CA*r^2;

if r == 0
    NA = 0.0;
else
    NA = NAr/r^2;
end
```

```
dCAdr = NA/(-De);
detadr = 3*CA*r^2/(CAs*R^3);
dydr = [dNArdr; dCAdr; detadr];
```

```
% -----
```

```
function bc = BCfun(ya, yb)
```

```
global CAs
```

```
bc = [ya(1); yb(2)-CAs; ya(3)];
```

计算结果 $\eta = 0.5630$ 。

【例 4-15】 多孔催化剂层中伴随有等温可逆反应的双组分气体扩散^[11]

在一蜂窝式反应器 (monolithic reactor) 的某一催化剂层中, 组分 A 和 B 之间发生气相催化可逆反应:



A 的反应速率为

$$r_A' = -k \left(C_A^2 - \frac{C_B}{K_c} \right) \quad (2)$$

式中, r_A' 为 $\text{mol}/(\text{cm}^3 \cdot \text{s})$; 速率常数 $k = 8 \times 10^4 \text{ cm}^3/(\text{mol} \cdot \text{s})$; 平衡常数 $K_c = 6 \times 10^5 \text{ cm}^3/\text{mol}$, 催化剂层厚度 $L = 0.2 \text{ cm}$ 。对于该催化剂层, 组分 A 在组分 B 中的扩散系数为 $D_e = 0.01 \text{ cm}^2/\text{s}$ 。

由于反应混合物只包含气体 A 和 B, 故只考虑双组分气体扩散。已知 A 和 B 的总浓度为 $C_T = 4 \times 10^{-5} \text{ mol}/\text{cm}^3$ 。试计算催化剂的有效因子。

数学模型

为了计算催化剂的有效因子, 必须进行物料衡算, 并考虑双组分的气体扩散。

在多孔催化剂层中 A 和 B 的物料衡算

从催化剂层顶部往下的 z 方向内对组分 A 进行物料衡算, 得:

$$\frac{dN_A}{dz} = r_A' = -k \left(C_A^2 - \frac{C_B}{K_c} \right) \quad (3)$$

$$\text{B.C.} \quad N_A|_{z=L} = 0 \quad (\text{多孔层底部没有传质通量}) \quad (4)$$

对于组分 B 的微分方程可用同样方法导出, 但更简单的方法是根据反应计量关系得到:

$$N_B = -\frac{1}{2} N_A \quad (5)$$

$$\text{B.C.} \quad N_B|_{z=L} = 0 \quad (\text{多孔层底部没有传质通量}) \quad (6)$$

双组分扩散的 Fick 定律

$$\text{在此双组分系统中 A 的扩散为: } N_A = -D_e \frac{dC_A}{dz} + x_A(N_A + N_B) \quad (7)$$

$$\text{其中催化剂层的有效扩散系数 } D_e \text{ 为: } D_e = \frac{D_{AB} \varepsilon_p \sigma}{\tau} \quad (8)$$

式中, D_{AB} 为双组分扩散系数; ε_p 为催化剂空隙率; σ 为收缩因子 (constriction factor); τ 为扭曲度。

把式 (5) 代入式 (7), 并考虑 x_A 的定义, 整理得:

$$\frac{dC_A}{dz} = \frac{\frac{C_A}{C_T} \left(\frac{N_A}{2} \right) - N_A}{D_e} \quad (9)$$

$$\text{I.C.} \quad C_A|_{z=0} = C_{AS} \quad (10)$$

对于组分 B 也可导出类似的微分方程, 但更简单的方法是, 根据总质量衡算得到:

$$C_B = C_T - C_A \quad (11)$$

$$\text{方程 (3) 可表达为关于 } C_A \text{ 的形式: } \frac{dN_A}{dz} = -k \left(C_A^2 - \frac{C_1 - C_A}{K_c} \right) \quad (12)$$

已知催化剂层表面处的反应物浓度为 $C_{AS} = 3 \times 10^{-5} \text{ mol/cm}^3$ 及 $C_{BS} = 1 \times 10^{-5} \text{ mol/cm}^3$ 。

因此, 以上微分方程 (9) 和 (12) 及边界条件 (10) 和 (4) 可构成 ODEs-BVP 问题。

催化剂层的有效因子 即催化剂层内的平均反应速率与催化剂层表面处的速率之比:

$$\eta = \frac{\int_0^L r_A' dz}{r_{AS}'} \quad (13)$$

$$\text{即 } \eta = \frac{\int_0^L \left(-k \left(C_A^2 - \frac{C_B}{K_c} \right) \right) dz}{-k \left(C_{AS}^2 - \frac{C_{BS}}{K_c} \right) L} = \frac{\int_0^L \left(C_A^2 - \frac{C_B}{K_c} \right) dz}{\left(C_{AS}^2 - \frac{C_{BS}}{K_c} \right) L}$$

$$\text{对 } z \text{ 求导, 得: } \frac{d\eta}{dz} = \frac{C_A^2 - \frac{C_B}{K_c}}{\left(C_{AS}^2 - \frac{C_{BS}}{K_c} \right) L} \quad (14)$$

$$\text{I.C.} \quad \eta|_{z=0} = 0 \quad (15)$$

而有效因子是 $z = L$ 处的 η 值。

所以, 由式 (9)、式 (12) 和式 (14) 构成 ODE 方程组, 其边界条件为式 (10)、式 (4) 和式 (15)。

程序说明 程序直接使用配置法函数 `bvp4c()` 求解此 ODE-BVPs 问题, 其中函数 `ODEs()` 定义方程式 (9)、式 (12) 和式 (14)。函数 `BCfun()` 则定义其边界条件式 (10)、式 (4) 和式 (15)。

程序清单 见光盘中的 *DiffReact_MonolithCat.m*。

计算结果 $\eta = 0.3007$ 。

【例 4-16】 氧扩散进入球形细胞中进行酶催化反应^[9]

数学模型 关于底物浓度的酶催化反应-扩散方程 (稳态) 为:

$$\nabla(D\nabla C) = g(C) \quad (1)$$

式中, D 为底物在细菌均匀分布的介质中的分子扩散系数; $g(C)$ 与反应速率成正比。现假设 D 为常数, 且反应速率用 Michaelis-Menten 方程, 得到如下无量纲反应-扩散方程:

$$(x^2 y')' = x^2 f(y), \quad 0 < x < 1 \quad (2)$$

$$\text{式中, } x = \frac{r}{R}, \quad y(x) = \frac{C(r)}{C_0}, \quad \varepsilon = \left(\frac{DC_0}{nqR^2} \right), \quad f(y) = \varepsilon^{-1} \frac{y(x)}{y(x) + k}, \quad k = \frac{k_m}{C_0} \quad (3)$$

$$\text{假设细胞膜具有无穷大的渗透系数, 则 } y(1) = 1 \quad (4)$$

$$\text{由于 } y(x) \text{ 关于 } x=0 \text{ 对称, 因此有 } y'(0) = 0 \quad (5)$$

已知 $\varepsilon = 0.1$, $k = 0.1$, 试模拟计算浓度分布。

程序说明 式 (2)、式 (4)、式 (5) 构成二阶 ODE-BVP 问题, 可降阶为一阶 ODE-BVPs

问题:

$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{2}{x}y_2 + f(y_1) \\ y_1(1) = 1, \quad y_2(0) = 0 \end{cases} \quad (6)$$

同前所述, 用函数 `bvpinit()` 和 `bvp4c()` 可求解 ODE-BVPs 问题 (6)。

注 若为 `bvpinit()` 提供的解的初值 (初始猜测值) 不当, 则会得出错误的结果。如本例中, 若 `bvpinit()` 的解的初值取为 `[1 0]`, 则结果 `y` 值出现负数, 没有物理意义。

程序清单 *EnzymeReactDiff.m*

```
function EnzymeReactDiff
clear all; clc
global epsilon k
a = 0; b = 1; epsilon = 0.1; k = 0.1; % 积分区间[a, b]
solinit = bvpinit(linspace(a, b, 50), [0 1]); % 以猜测值 y1(x)=0, y2(x)=1 对解进行初始化
sol = bvp4c(@ODEfun, @BCfun, solinit);
x = [0:0.2:1]; y = deval(sol, x);
fprintf('Results:\n \tx\t\ty\n'), fprintf('\t%.4f \t%.4f\n', [x; y(1, :)])

% 结果显示
x4plot = linspace(a, b, 200); y4plot = deval(sol, x4plot);
plot(x4plot, y4plot(1, :)), xlabel('x'), ylabel('y')

% -----
function dydx = ODEfun(x, y)
global epsilon k
fy1 = 1/epsilon * y(1)/(y(1)+k);
dydx(1) = y(2);
if x==0
    dydx(2) = 0;
else
    dydx(2) = -2*y(2)/x + fy1;
end
dydx = dydx';

% -----
function bc = BCfun(ya, yb)
bc = [yb(1)-1; ya(2)];
```

计算结果 浓度分布见图 4-18。

x	0	0.2000	0.4000	0.6000	0.8000	1.0000
y	0.0228	0.0368	0.0987	0.2569	0.5523	1.0000

【例 4-17】通过对药丸包衣的溶解控制药物释放——固体在溶液中的非稳态溶解^[11]

包含某种药的药丸，其内核是纯药 D，外部由包衣 A 包围，以控制药的释放。在胃里面，外层包衣和药的溶解速率不同。设 C_{AS} 为胃中包衣的浓度 (mg/cm^3)； C_{DS} 为胃中药的浓度 (mg/cm^3)； C_{DB} 为体内（血液中）药的浓度 (mg/kg)。现有以下三种不同的药丸：

药丸 1: $D_{A1} = 5 \text{ mm}$, $D_{D1} = 3 \text{ mm}$

药丸 2: $D_{A2} = 4 \text{ mm}$, $D_{D2} = 3 \text{ mm}$

药丸 3: $D_{A3} = 3.5 \text{ mm}$, $D_{D3} = 3 \text{ mm}$

其中， D_{Ai} 为药丸 i 的包衣 A 的直径， D_{Di} 为药丸 i 的纯药 D 的直径。

其他数据：每粒药丸内核中的药量 = 20 mg，内层和外层的总密度 = $1414.7 \text{ mg}/\text{cm}^3$ ，胃中包衣的溶解度 $S_A = 1.0 \text{ mg}/\text{cm}^3$ ，胃中内层药的溶解度 $S_D = 0.4 \text{ mg}/\text{cm}^3$ ，胃中液体的体积 $V = 1.2 \text{ L}$ ，胃中停留时间 $\tau = V/v_0 = 4 \text{ h}$ ，人的典型体重 $w = 75 \text{ kg}$ ，Sherwood 数 $Sh = k_L \frac{D_p}{D_{AB}} = 2$ ，A 和 D

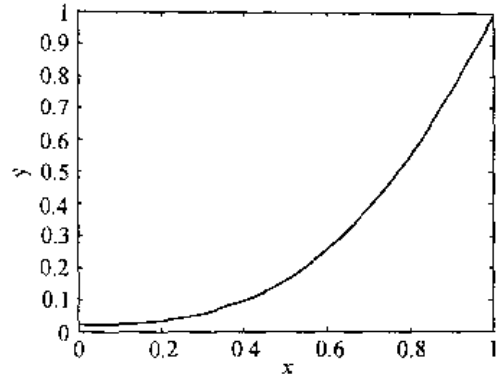


图 4-18 浓度分布

在胃中的有效扩散系数 $D_A = D_D = 0.6 \text{ cm}^2/\text{min}$ 。

现在一个人同时服下三颗药丸（药丸 1、2、3 各一颗），假设其胃中混合良好，且在溶解期间，药丸一直保留在胃中。求：

- ① 绘制服药后 2.5 h 内三颗药丸直径随时间的变化。
- ② 绘制服药后 2.5 h 内胃中药物浓度 C_{AS} 和 C_{BS} 随时间的变化。

数学模型

需要对三颗药丸建立微分方程，并同时求解。只有当外层溶解后，药材才开始释放。药的溶解过程与气体的升华过程相似。

对药丸进行物料衡算

对药丸 1

$$\text{在 } D_1 > D_{D1} \text{ (外层) 处,} \quad \frac{dD_1}{dt} = -\frac{2k_{L1}}{\rho}(S_A - C_{AS}) \quad (1)$$

$$\text{I.C.} \quad D_1|_{t=0} = D_{A1} \quad (2)$$

$$\text{在 } 10^{-5} \leq D_1 \leq D_{D1} \text{ (纯药 D) 处,} \quad \frac{dD_1}{dt} = -\frac{2k_{L1}}{\rho}(S_D - C_{DS}) \quad (3)$$

$$\text{在 } D_1 \leq 10^{-5} \text{ (全部溶解) 处,} \quad \frac{dD_1}{dt} = 0 \quad (4)$$

$$\text{其中, 药物 1 的传质系数 } k_{L1} \text{ 取决于直径 } D_1: k_{L1} = \frac{2(0.6)}{D_1} \quad (5)$$

描述药丸 2 和药丸 3 的颗粒直径 D_2 和 D_3 的微分方程，与药丸 1 完全类似，此处不再列出。

对胃中组分 A 进行物料衡算

$$\begin{aligned} \frac{dC_{AS}}{dt} = & \frac{1}{V} [S_{W1}k_{L1}(S_A - C_{AS})\pi D_1^2 + S_{W2}k_{L2}(S_A - C_{AS})\pi D_2^2 \\ & + S_{W3}k_{L3}(S_A - C_{AS})\pi D_3^2] - \frac{C_{AS}}{\tau} \end{aligned} \quad (6)$$

对胃中组分 D 进行物料衡算

$$\frac{dC_{DS}}{dt} = \frac{1}{V} [(1-S_{w1})k_{L1}(S_D - C_{DS})\pi D_1^2 + (1-S_{w2})k_{L2}(S_D - C_{DS})\pi D_2^2 + (1-S_{w3})k_{L3}(S_D - C_{DS})\pi D_3^2] - \frac{C_{DS}}{\tau} \quad (7)$$

程序说明 本例是同时包括传质和反应的 ODE-IVP 问题，程序用 ode45() 求解，其中 ODEs() 定义 ODE 方程组。像式 (1)、式 (3)、式 (4) 这样的方程，程序用 if...else...end 处理。

程序清单 见光盘中的 DrugDelivery.m。

计算结果 见图 4-19 和图 4-20。

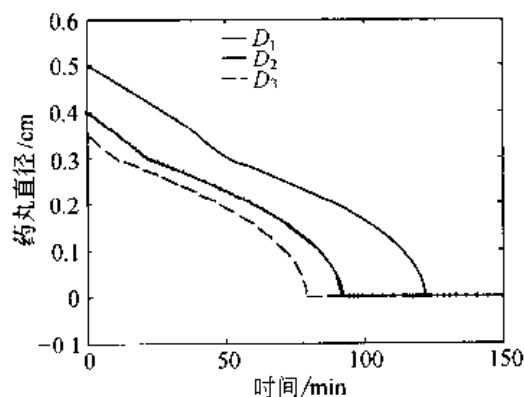


图 4-19 三颗药丸在溶解过程中的直径变化

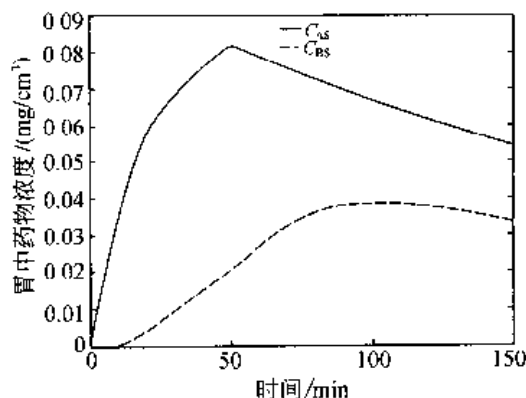


图 4-20 胃中药物浓度的动态变化

4.7 传热过程

可逆气固化学热泵制冷系统的关键设备是气-固反应器，但该系统涉及许多传热问题，所以将它归入这一节作为传热过程的例子。普通传热设备的动态模拟与此制冷系统类似。

【例 4-18】可逆气固化学热泵制冷系统的动态模拟^[12, 13]

以太阳能、工业余热等低温热源为驱动能源的化学热泵制冷系统，具有节能降耗和绿色环保等诸多优点，可广泛应用于化工、炼油等

工业领域的制冷，也可用于新型的无噪音、无公害的汽车空调、大楼空调及太阳能制冰机等。

气-固化学热泵制冷系统如图 4-21 所示，本系统包括两个装入相同 SrCl_2 反应材料的反应器（吸附/解吸器）R1 和 R2、一个蒸发器(E) 和一个冷凝器(C)。其中，解吸器与热油换热器(EXh)相连，热油为解吸器提供热量，而吸附器与冷油换热器(EXc)相连，反应床层由冷油冷却。该系统操作是一个周期性连续循环，每个反应器交替地吸附（冷却）和解吸（加热）氨，可进行拟连续制冷。下面模拟计算吸附/解吸过程的动态行为。

数学模型 模型假设：

(1) 由实验测知径向压差接近于零，因此

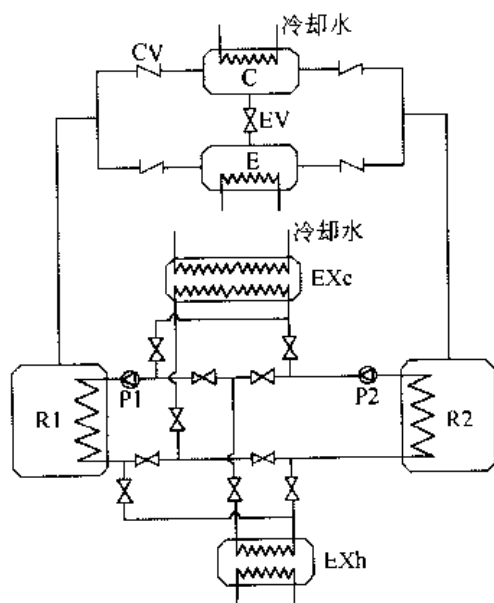


图 4-21 连续回热型双反应器化学热泵的系统流程

可假设反应床层内各处压力相同；

(2) 吸附材料导热性能很好，可假设反应床层内各处温度均匀一致；

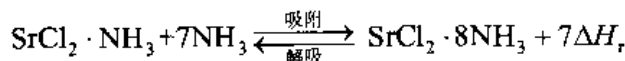
(3) 忽略吸附床的轴向导热；

(4) 在吸附/解吸过程中气体流率很低，可忽略其对流传热；

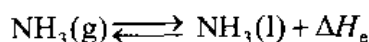
(5) 换热器、蒸发器和冷凝器的传热能力与吸附/解吸器相匹配。

化学平衡方程

$\text{SrCl}_2/\text{NH}_3$ 可逆气-固化学反应式为：



蒸发器/冷凝器中制冷剂的气-液相平衡为：



可逆化学平衡和气-液平衡均遵循 Clapeyron 方程：

$$\ln p_{\text{eq}} = -\frac{4983.28}{T} + 27.5100 \quad (\text{SrCl}_2/\text{NH}_3 \text{ 可逆气-固化学平衡}) \quad (1a)$$

$$\ln p_{\text{eq}} = -\frac{2764.4}{T} + 23.0269 \quad (\text{氨的气-液平衡}) \quad (1b)$$

可逆气-固化学反应动力学模型

$\text{SrCl}_2/\text{NH}_3$ 可逆气-固化学反应动力学方程为：

$$\frac{dX}{dt} = k_{0a} \exp\left(-\frac{E_a}{RT}\right) \cdot [1-X]^{M_a} \cdot \frac{p_c - p_{\text{eq}}(T)}{p_{\text{eq}}(T)} \quad (\text{吸附}) \quad (2a)$$

$$\frac{dX}{dt} = k_{0d} \exp\left(-\frac{E_d}{RT}\right) \cdot [1-X]^{M_d} \cdot \frac{p_{\text{eq}}(T) - p_c}{p_{\text{eq}}(T)} \quad (\text{解吸}) \quad (2b)$$

式中， X 为反应进度； T 为床层温度； k_{0a} , E_a , M_a 及 k_{0d} , E_d , M_d 为动力学参数。

床层热量平衡方程：

$$m_s c_{ps}(X) \frac{dT_s}{dt} = h_{st}(T_t - T_s) \pm 7N_s \Delta H_r \cdot \frac{dX}{dt} \quad (3)$$

式中 c_{ps} 为单位体积床层的比热容 ($\text{J} \cdot \text{kg}^{-1} \cdot \text{K}^{-1}$)，其计算公式如下：

$$c_{ps} = \rho_s \times w_{gr} \times c_{pgr} + c_{ps1} \times M_{s1} + c_{ps2} \times M_{s2} \quad (3a)$$

$$M_{s1} = \rho_s \times (1 - w_{gr}) \times (1 + M_g / M_s) \times (1 - r) \quad (3b)$$

$$M_{s2} = \rho_s \times (1 - w_{gr}) \times (1 + 8 \times M_g / M_s) \times r \quad (3c)$$

$r = X$ (吸附时)； $r = 1 - X$ (解吸时)。

反应管壁的热量平衡方程

$$m_t c_{pt} \frac{dT_t}{dt} = h_{st} A_{st} (T_s - T_t) - h_{tf} A_{tf} (T_t - T_f) \quad (4)$$

夹套内载热流体的热量平衡方程

$$m_f c_{pf} \frac{dT_f}{dt} = h_{tf} A_{tf} (T_t - T_f) - h_{fe} A_{fe} (T_f - T_a) - F_f \rho_f c_{pf} (T_f - T_{fin}) \quad (5)$$

模型参数如表 4-3 所示。

表 4-3 模型参数

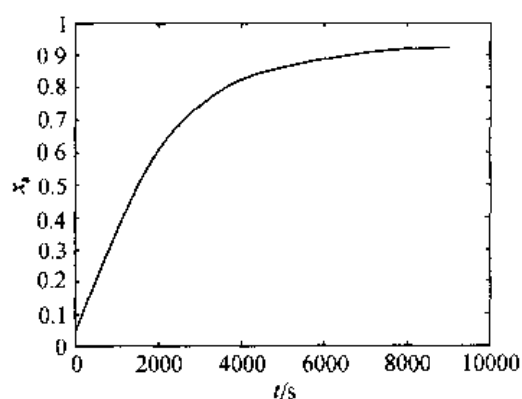
反应器结构		系统物性		设计	
参 数	数 值	参 数	数 值	参 数	数 值
D/m	0.150	w_{gr}	0.30	M_t/kg	25
D_0/m	0.006	$c_{pi}/J \cdot kg^{-1} \cdot K^{-1}$	50	m_t/kg	1.390
H/m	0.100	$c_{ps1}/J \cdot kg^{-1} \cdot K^{-1}$	697	m_s/g	1037
e_1/m	0.0045	$c_{ps2}/J \cdot kg^{-1} \cdot K^{-1}$	1480	$F_0/m^3 \cdot s^{-1}$	80×10^{-6}
e_2/m	0.012	$c_{p0}/J \cdot kg^{-1} \cdot K^{-1}$	674	$h_{st}/W \cdot m^{-2} \cdot K^{-1}$	500
		$c_{p0}/J \cdot kg^{-1} \cdot K^{-1}$	2.4	$h_{01}/W \cdot m^{-2} \cdot K^{-1}$	692
		$\rho_0/kg \cdot m^{-3}$	870	$h_{02}/W \cdot m^{-2} \cdot K^{-1}$	2
		$M_s/g \cdot mol^{-1}$	158.526		
		$M_g/g \cdot mol^{-1}$	17		
		$\Delta H_0/J \cdot mol^{-1}$	23366		
		$\Delta H_r/J \cdot mol^{-1}$	41431		

程序说明 Equations()定义方程组 (2a)、(2b)、(3) ~ (5)。先用 ode45()求解, 发现很久不能算出结果, 说明很可能是刚性方程组, 这时, 在命令窗口同时按<Ctrl>和<Break>键中断程序, 再将 ode45()改为 ode23s()即求得结果, 说明此问题确实是刚性问题。

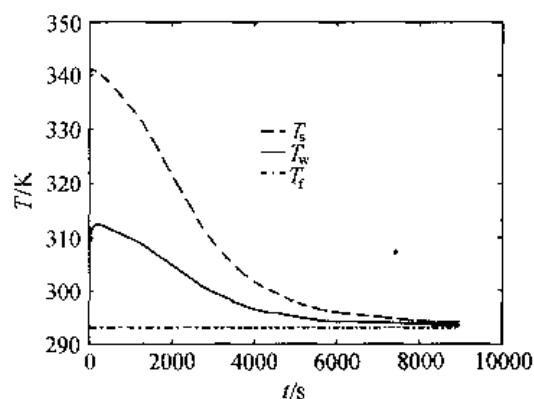
程序稍加修改补充, 还可模拟在吸附过程中冷油对总反应进度、床层温度的影响, 也可对解吸过程进行类似于吸附过程那样的详细研究。

程序清单 见光盘中的 ChemHeatPump.m。

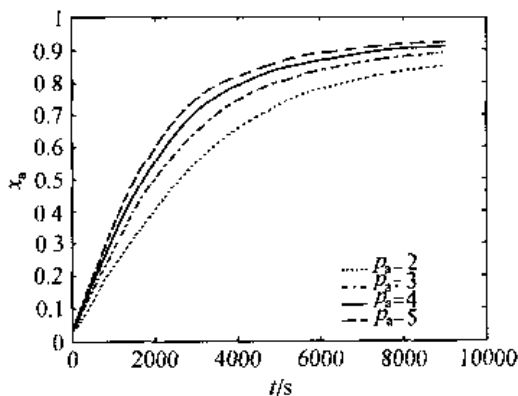
计算结果 吸附过程的结果如图 4-22 所示, 解附过程的结果从略。



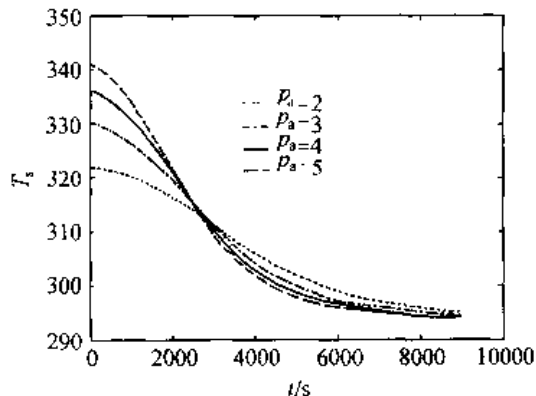
(a) 吸附过程总反应进度的动态变化



(b) 吸附过程床层温度、壁温、冷油温度的变化



(c) 吸附压力对总反应进度的影响



(d) 吸附压力对床层温度的影响

图 4-22 吸附过程

符号说明

c_{pt} ——反应器质量热容, $\text{J}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$	M_t ——反应器质量 (包括反应管及上下盖), kg
c_{ps1} —— $\text{SrCl}_2\cdot\text{NH}_3$ 的比热容, $\text{J}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$	m_t —— SrCl_2 反应材料的总质量, kg
c_{ps2} —— $\text{SrCl}_2\cdot 8\text{NH}_3$ 的比热容, $\text{J}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$	$N(\text{SrCl}_2)$ ——反应盐 SrCl_2 的物质的量 (摩尔数), mol
c_{pg} ——可膨胀石墨的比热容, $\text{J}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$	p_a ——吸附压力, Pa
c_{pf} ——载热介质的比热容, $\text{kJ}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$	p_d ——解吸压力, Pa
D ——吸附器内径, m	$p_{ea}(T)$ ——对应于温度 T 的吸附平衡压力, Pa
D_0 ——气体扩散器 (中心孔) 的直径, m	$p_{ed}(T)$ ——对应于温度 T 的解吸平衡压力, Pa
H ——反应材料的高度, m	p_{eq} ——应于温度 $T(\text{K})$ 的平衡压力, Pa
e_1 ——反应管壁厚, m	R ——理想气体常数
e_2 ——火套载热流体流通截面的宽度, m	P_w ——单位质量反应材料的瞬时制冷功率, $\text{W}\cdot\text{kg}^{-1}$
E_a ——吸附活化能, $\text{J}\cdot\text{mol}^{-1}$	T_{a0} 、 T_{d0} ——分别为吸附床和解吸床的初始温度, K
E_d ——解吸活化能, $\text{J}\cdot\text{mol}^{-1}$	T_{ea} 、 T_{ed} ——分别对应于 p_a 和 p_d 的平衡温度, K
h_{fe} 、 h_{ft} 、 h_{st} ——分别是载热流体与环境、载热流体与反应器壁和反应床层与反应器壁之间的传热系数, $\text{W}\cdot\text{m}^{-2}\cdot\text{K}^{-1}$	T_f ——载热流体的平均温度, K
k_{0a} ——吸附动力学 Arrhenius 项的指前因子, s^{-1}	T_{fa} 、 T_{fd} ——分别为吸附器和解吸器的夹套套内流体的平均温度, K
k_{0d} ——解吸动力学 Arrhenius 项的指前因子, s^{-1}	T_{lc} 、 T_{th} ——分别为冷、热油温度, K
M_a ——吸附动力学幂指数	T_s ——反应器床层平均温度, K
M_d ——解吸动力学幂指数	T_t ——反应管 (壁) 平均温度, K
F_f ——载热介质的体积流量, $\text{m}^3\cdot\text{s}^{-1}$	T_{ta} 、 T_{td} ——分别为吸附器和解吸器的管壁平均温度, K
$M(\text{NH}_3)$ ——氨气的摩尔质量, $\text{g}\cdot\text{mol}^{-1}$	w_{gr} ——可膨胀石墨的质量分数
$M(\text{SrCl}_2)$ ——反应盐 SrCl_2 的摩尔质量, $\text{g}\cdot\text{mol}^{-1}$	X ——反应进度
$m(\text{SrCl}_2)$ —— SrCl_2 反应材料中 SrCl_2 盐的质量	ρ_c ——载热介质 (导热油) 的密度, $\text{kg}\cdot\text{m}^{-3}$
M_{s1} ——单位体积反应材料中 $\text{SrCl}_2\cdot\text{NH}_3$ 的质量, $\text{kg}\cdot\text{m}^{-3}$	ρ_s ——单位体积反应材料所含的反应盐密度, $\text{kg}\cdot\text{m}^{-3}$
M_{s2} ——单位体积反应材料中 $\text{SrCl}_2\cdot 8\text{NH}_3$ 的质量, $\text{kg}\cdot\text{m}^{-3}$	ΔH_e ——氨的蒸发潜热, $\text{J}\cdot\text{mol}^{-1}$
	ΔH_r ——对 SrCl_2 的反应热, $\text{J}\cdot\text{mol}^{-1}$

4.8 流体流动

【例 4-19】水平管中不可压缩牛顿流体的层流流动^[14]

求解动量守恒方程以获得剪应力分布、剪切速率分布和平均速率。

数学模型

$$\frac{d}{dr}(r\tau_{rx}) = \left(\frac{\Delta p}{L}\right)r \quad (1)$$

$$\text{B.C.} \quad \tau_{rx}|_{r=0} = 0 \quad (2a)$$

$$(r\tau_{rx})|_{r=0} = 0 \quad (2b)$$

式中, r 为管任意半径的位置, m ; τ_{rx} 为管半径 r 处的剪应力, $\text{kg}/(\text{m}\cdot\text{s}^2)$; Δp 为压降, Pa ; L 为长度, m 。

对于牛顿型流体, 剪应力 (或动量通量) 为:

$$\tau_{rx} = -\mu \frac{du_x}{dr} \quad (3a)$$

$$\text{即} \quad \frac{du_x}{dr} = -\frac{\tau_{rx}}{\mu} \quad (3b)$$

$$\text{B.C.} \quad u_x|_{r=R} = 0 \quad (4)$$

$$\text{平均剪切速率:} \quad u_m = \frac{1}{\pi R^2} \int_0^R u 2\pi r dr \quad (5a)$$

$$\text{两边同时对 } r \text{ 求导, 得} \quad \frac{du_m}{dr} = \frac{2ur}{R^2} \quad (5b)$$

$$\text{I.C.} \quad u_m|_{r=0} = 0 \quad (6)$$

$$\text{由此, 得到 ODE-BVPs:} \quad \frac{d}{dr}(r\tau_{rx}) = \left(\frac{\Delta p}{L}\right)r \quad (1)$$

$$\frac{du_x}{dr} = -\frac{\tau_{rx}}{\mu} \quad (3b)$$

$$\frac{du_m}{dr} = \frac{2ur}{R^2} \quad (5b)$$

$$(r\tau_{rx})|_{r=0} = 0, \quad u_x|_{r=R} = 0, \quad u_m|_{r=0} = 0 \quad (2b)$$

程序说明 程序用 bvp4c() 求解由式 (1)、(3b)、(5b) 和 (2b) 组成 ODE-BVPs 问题。

程序清单 程序见光盘中的 *NewtonFluidFlow.m*。

计算结果 剪应力分布和剪切速率分布如图 4-23 和 4-24。平均剪切速率为 $u_m = 0.6042 \text{ m/s}$ 。

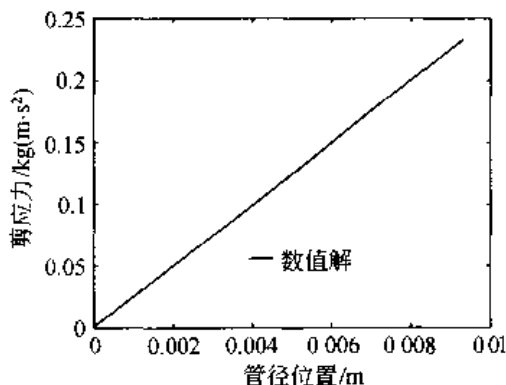


图 4-23 剪应力分布

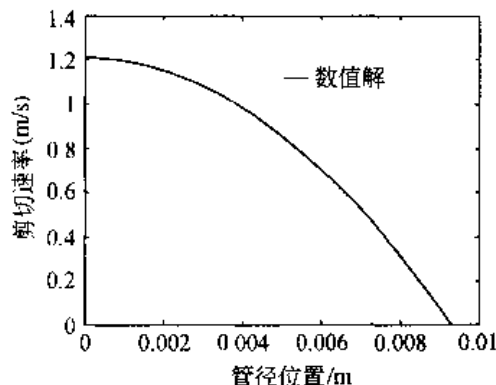


图 4-24 剪切速率分布

4.9 生化反应

【例 4-20】厌氧间歇发酵动态模拟^[15]

使用 *Saccharomyces cerevisiae* (一种干酵母) 厌氧间歇发酵葡萄糖生成乙醇和副产物 CO_2 。物料包含 10% 的葡萄糖。接种体是 0.0005 (质量比)。在 14% 乙醇时由于产品抑制使细胞停止生长。假设 $k_d = 0$ 但当底物完全消耗时, 开始消耗细胞质。试模拟计算从发酵开始各组分的浓度动态变化。已知初始浓度为: $t = 0$, $X_0 = 0.0005$, $S_0 = 0.04$, $p_0 = 0$ 。参数如下:

$$u_{\max} = 0.5 \text{ h}^{-1}, \quad K_s = 0.001, \quad m = 1, \quad M_s = 0.008 \text{ h}^{-1}, \quad \hat{Y}_{x/s} = 2, \quad Y_{x/s} = 1$$

数学模型

质量平衡方程

细胞的质量平衡方程为
$$\frac{dX}{dt} = R_X \quad (1)$$

底物的质量平衡方程为
$$\frac{dS}{dt} = R_S \quad (2)$$

产物的质量平衡方程为
$$\frac{dp}{dt} = R_p \quad (3)$$

动力学方程

有抑制时细胞生长的速率方程为
$$R_X = \mu_{\max} X \left(\frac{S}{K_S + S} \right) \left(1 - \frac{p}{p_{\max}} \right)^m \quad (4)$$

底物消耗的速率方程为
$$R_S = -\frac{R_X}{Y_{X/S}} - M_S X \quad (5)$$

产物的生成速率方程为
$$R_p = \frac{R_X}{Y_{X/S}} - \frac{R_X}{\hat{Y}_{X/S}} + M_S X \quad (6)$$

程序说明 这是常微分方程组的初值问题 (ODE-IVPs)，程序用 ode45() 求解。

程序清单 *BatchBR_CellCult.m*

```
function BatchBR_CellCult
clear all; clc
global umax KS pmax m YX2S YX2Shat MS

% 已知参数
umax = 0.5; pmax = 0.109; KS = 0.001; m = 1; YX2Shat = 2; YX2S = 1;
MS = 0.008;

% 已知初始浓度
X0 = 0.0005; S0 = 0.04; p0 = 0

% 求解 ODE-IVPs
tspan = [0 15]; y0 = [X0 S0 p0];
[t y] = ode45(@ModelEqs, tspan, y0)

% 绘图
plot(t, y), xlabel('反应时间, h'), ylabel('量纲为一的浓度, 质量分数')
str = {'活细胞', '葡萄糖', '乙醇'};
for i = 1:3, gtext(str{i}), end

% -----
function dydt = ModelEqs(t, y, k) % 模型方程
global umax KS pmax m YX2S YX2Shat MS
X = y(1); S = y(2); p = y(3);
RX = umax * X * S / (KS + S) * (1 - p / pmax)^m;
```

$$R_S = -R_X/Y_{X2S} - M_S \cdot X;$$

$$R_p = R_X/Y_{X2S} - R_X/Y_{X2S} \text{hat} + M_S \cdot X;$$

$$dydt = [R_X; R_S; R_p];$$

计算结果 从发酵开始各组分的浓度动态变化如图 4-25 所示。

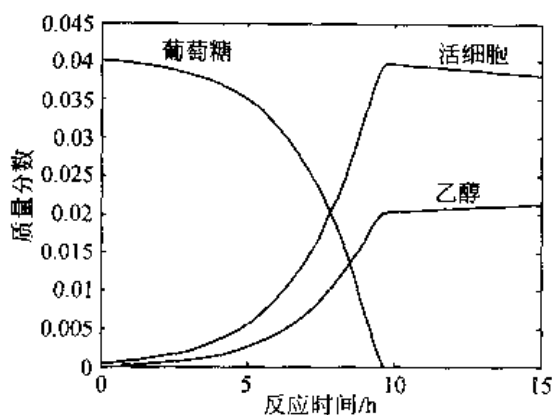


图 4-25 厌氧间歇发酵的浓度动态变化

【例 4-21】生化反应器模拟计算^[16]

在一个连续搅拌槽式发酵罐 (CSTF) 中进行生化反应, x_1 为生物量的浓度 (单位体积所含的细胞质量), x_2 为酶解物 (底物 substrate) 的浓度 (单位体积所含的酶解物质量), x_{f1} 、 x_{f2} 分别为进料中所含的生物量和酶解物的浓度, F 为料液的体积流量, V 为工作体积。已知数据为: $D = 0.3$, $\mu_{\max} = 0.53$, $Y = 0.4$, $k_m = 0.12$, $x_{f1} = 0$, $x_{f2} = 4.0$, $k_1 = 0.4545$ 。

(1) 稳态模拟 计算稳态时生物量和底物的浓度;

(2) 动态模拟 计算生物量和底物浓度随时间的变化。

数学模型

对生物量进行物料衡算:

$$\frac{dVx_1}{dt} = Fx_{f1} - Fx_1 + Vr_1 \quad (1)$$

对酶解物进行物料衡算:

$$\frac{dVx_2}{dt} = Fx_{f2} - Fx_2 + Vr_2 \quad (2)$$

反应速率 (单位时间单位体积所产生的细胞量): $r_1 = \mu x_1$ (3)

式中, μ 为生长速率系数 (specific growth rate coefficient), 单位为 s^{-1} 。通常 μ 并非是常数, 而是与酶解物的浓度有关。最常见的表达式有 Monod 和 Substrate inhibition:

$$\mu = \frac{\mu_{\max} x_2}{k_m + x_2} \quad (\text{Monod}) \quad (4)$$

$$\mu = \frac{\mu_{\max} x_2}{k_m + x_2 + k_1 x_2^2} \quad (\text{Substrate inhibition}) \quad (5)$$

收率:

$$Y = \frac{\text{产生的细胞量}}{\text{酶解物的消耗量}} = \frac{r_1}{r_2} \quad (6)$$

由式(6)得 r_2 , 并和式(3)的 r_1 一起代入方程(1)和(2), 整理得(假设 $x_{f1} = 0$)

$$\frac{dx_1}{dt} = (\mu - D)x_1 \quad (7)$$

$$\frac{dx_2}{dt} = D(x_{f2} - x_2) - \frac{\mu x_1}{Y} \quad (8)$$

式中 $D = F/V$ (9)

方程(7)、(8)就是主要的动态模型方程组, 其初始条件为:

$$t = 0, x_1 = 1, x_2 = 1 \quad (10)$$

令 $dx_1/dt = 0$, $dx_2/dt = 0$, 则由式 (7) 和 (8) 可得到稳态模型:

$$(\mu - D)x_1 = 0 \quad (11)$$

$$D(x_{f2} - x_2) - \frac{\mu x_1}{Y} = 0 \quad (12)$$

程序说明 用 `fsolve()` 求解稳态模型 (11) 和 (12)，用 `ode45()` 求解由式 (7)、(8) 和 (10) 组成 ODE-IVPs 问题 (动态模型)。

程序清单 见光盘中的 *BioReactCSTF.m*。

计算结果 生物量和底物浓度的稳态值分别是 0.995 和 1.512，而其动态变化如图 4-26 所示。

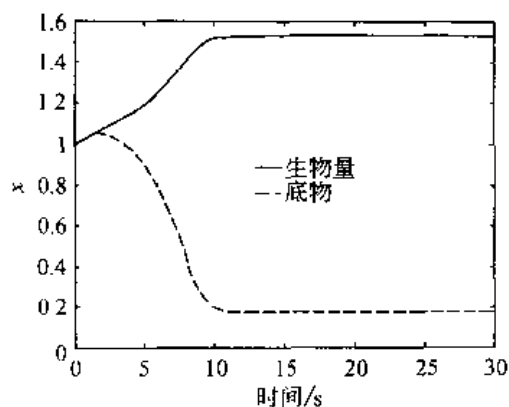


图 4-26 生物量和底物浓度随时间的变化

符号说明

D 稀释速度 ($D = F/V$), s^{-1}
 F 料液的体积流量, L/s
 k_1 动力学参数
 k_m 动力学参数
 r 反应速率
 t 时间, s

下标

1、2 分别表示生物量和底物

V 反应器中的料液体积, L
 x 浓度
 Y 收率
 μ 生长速率系数, s^{-1}
 μ_{max} 动力学参数

f 进料

4.10 分离-反应一体化设备

膜反应器和反应精馏是分离-反应一体化设备的典型例子。在膜反应器中，反应产物在生成的同时，部分或全部脱离反应区，使得化学平衡不断地向生成物方向移动。因此，在膜反应器中进行的反应，其转化率将高于热力学平衡转化率(传统反应器的理想转化率)。在反应精馏装置中同时进行精馏和反应，可大大降低投资费用和操作费用。膜反应器和反应精馏的数学模型都是一组 ODE 方程，容易数值求解。因篇幅所限，本节只介绍膜反应器的模拟，反应精馏将不作介绍，读者可参考 Fogler^[10]。

【例 4-22】膜催化反应器的模拟计算^[10]

如图 4-27 所示，在一容积为 500 L 的 IMRCF 型膜催化反应器中进行丙烷脱氢反应： $C_3H_8 = C_3H_6 + H_2$ ，用符号表示为 $A = B + C$ 。反应发生在 IMRCF 的催化剂侧，其中 B (H_2) 可透过膜，而 A 和 C 则不能透过。227℃ 时该反应的平衡常数相当小： $K_C = 0.05 \text{ mol/dm}^3$ 。纯气体 A 在 227℃ 和 830.865 kPa (8.2 atm) 下以 10 mol/min 的流率进入反应器。

假设每单位体积反应器扩散出反应器 B 的流率 R_B 与 B 的浓度成正比，即 $R_B = k_c C_B$ 。

虽然此反应是气-固催化反应，我们将使用催化剂的堆密度 (回想 $-r_A = -r'_A \rho_b$)，以便写出基于反应器体积而不是催化剂重量的衡算方程。催化剂堆密度 $\rho_b = 1.5 \text{ g/cm}^3$ ，内装催化剂颗粒的反应管内径为 2 cm，反应速率常数 k 和传递系数 k_c 分别为 $k = 0.7 \text{ min}^{-1}$ 和 $k_c = 0.2 \text{ min}^{-1}$ 。

(a) 试对组分 A、B 和 C 分别进行微分摩尔衡算，导出一组需联立求解的微分方程。

(b) 标绘各组分摩尔流率作为空时的函数关系图。

数学模型 以反应器体积为自变量，对图 4-28 所示的微元体 ΔV 进行摩尔衡算。

(1) 摩尔衡算

对催化床层内组分 A 的衡算

$$\begin{array}{ccccccc} \text{[流入]} & - & \text{[流出]} & + & \text{[产生]} & = & \text{[积累]} \\ F_A|_V & - & F_A|_{V+\Delta V} & + & r_A \Delta V & = & 0 \end{array}$$

各项除以 ΔV ，并取 $\Delta V \rightarrow 0$ 的极限，得： $\frac{dF_A}{dV} = r_A$ (1)

对催化床层内组分 B 的衡算

$$\begin{array}{ccccccc} \text{[流入]} & - & \text{[流出]} & - & \text{[扩散出]} & + & \text{[产生]} & = & \text{[积累]} \\ F_B|_V & - & F_B|_{V+\Delta V} & - & R_B \Delta V & + & R_B \Delta V & = & 0 \end{array}$$

各项除以 ΔV ，并取 $\Delta V \rightarrow 0$ 的极限，得 $\frac{dF_B}{dV} = r_B - R_B$ (2)

组分 C 的摩尔衡算与组分 A 相同，得到的方程为

$$\frac{dF_C}{dV} = r_C \quad (3)$$

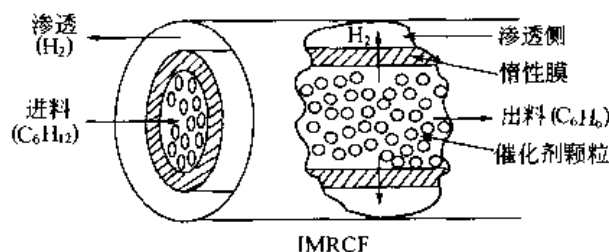


图 4-27 IMRCF 膜反应器

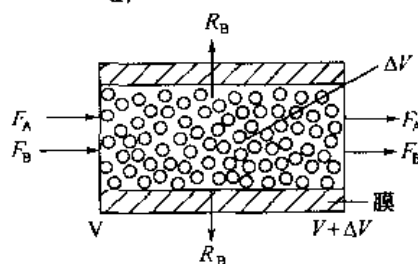


图 4-28 摩尔衡算示意图

(2) 速率方程
$$-r_A = k(C_A - \frac{C_B C_C}{K_C}) \quad (4)$$

$$r_B = -r_A$$

$$r_C = -r_A$$

(3) 由反应器侧面传出

前面已假定

$$R_B = k_c C_B \quad (5)$$

一般来说，传递系数 k_c 可以是膜和流体的性质、流体速度、管径等的函数，但本例假定 B 扩散出反应器的主要阻力在于膜本身，因此 k_c 可视为常数。

(4) 化学计量关系

对于温度和压力均恒定的情况 ($T = T_0$, $p = p_0$)，有

$$C_A = C_{T0} \frac{F_A}{F_T} \quad (6)$$

$$C_B = C_{T0} \frac{F_B}{F_T} \quad (7)$$

$$C_C = C_{T0} \frac{F_C}{F_T} \quad (8)$$

$$F_T = F_A + F_B + F_C \quad (9)$$

$$-r_A = r_B = r_C \quad (10)$$

(5) 结合上述各方程总结如下

$$\frac{dF_A}{dV} = r_A$$

$$\frac{dF_B}{dV} = -r_A - k_c C_{T0} \left(\frac{F_B}{F_T} \right)$$

$$\frac{dF_C}{dV} = -r_A$$

$$-r_A = kC_{T0} \left[\frac{F_A}{F_T} - \frac{C_{T0}}{K_C} \left(\frac{F_B}{F_T} \right) \left(\frac{F_C}{F_T} \right) \right]$$

$$F_T = F_A + F_B + F_C$$

程序说明 题目要求标绘各组分摩尔流率-空时关系图, 考虑空时与体积存在对应关系, 这里标绘各组分摩尔流率-体积关系图。

程序清单 见光盘中的 *MembraneReactor.m*。

计算结果 各组分摩尔流率随体积的变化如图 4-29 所示。

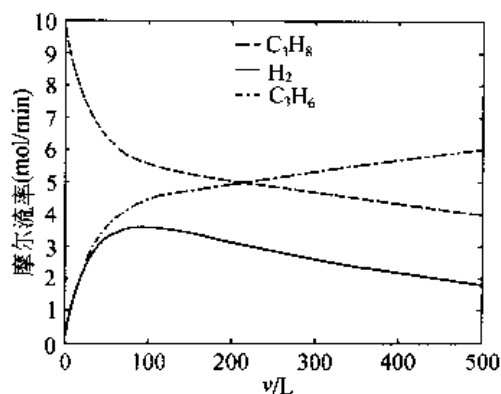


图 4-29 各组分摩尔流率随体积的变化

4.11 过程控制

【例 4-23】非等温 CSTR^[2]

如图 4-30 所示, 在一个非等温 CSTR 反应器中进行反应: $A \xrightarrow{k} B$, 假设流量和持液量 (液体容积 V) 都随时间变化, 可用一个液位比例控制器操纵离开反应器的液体流量 F , 即流量与反应器液体容积成线性关系:

$$F = 40 - 10(48 - V) \quad (1)$$

用第二个控制器操纵进入夹套的冷却水流量 F_J , 它与反应器温度成正比:

$$F_J = 49.9 - Kc(600 - T) \quad (2)$$

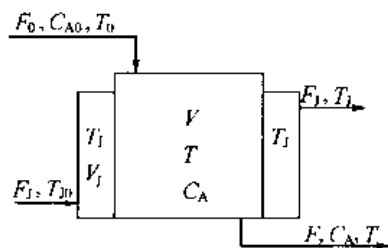


图 4-30 非等温 CSTR

假设冷却夹套中持液量恒定、混合完全。系统原先处于稳态过程 ($t = 0$ 时刻之前), 在 $t = 0$ 时刻, 入口处浓度 $CA0$ 和/或流量 $F0$ 出现扰动 (阶跃变化), 试模拟计算在控制器的调节作用下, 各参数随时间的动态变化。已知数据如下。

稳态值●

$$CA0 = 0.50 \text{ lb} \cdot \text{mol} / \text{ft}^3$$

$$F = 40 \text{ ft}^3 / \text{h}$$

$$T = 600 \text{ } ^\circ\text{R}$$

$$T_J = 594.6 \text{ } ^\circ\text{R}$$

$$CA = 0.245 \text{ lb} \cdot \text{mol} / \text{ft}^3$$

$$F_J = 49.9 \text{ ft}^3 / \text{h}$$

$$T_0 = 530 \text{ } ^\circ\text{R}$$

$$V = 48 \text{ ft}^3$$

参数值●

$$A_H = 250 \text{ ft}^2$$

$$c_p = 0.75 \text{ Btu} / (\text{lb}_m \cdot ^\circ\text{R})$$

$$Kc = 4 \text{ (ft}^3 / \text{h)} / ^\circ\text{R}$$

$$T_{J0} = 530 \text{ } ^\circ\text{R}$$

$$U = 150 \text{ Btu} / (\text{h ft}^2 \cdot ^\circ\text{R})$$

$$k_0 = 7.08 \times 10^{10} \text{ h}^{-1}$$

$$\rho = 50 \text{ lb}_m / \text{ft}^3$$

$$c_{pJ} = 1.0 \text{ Btu} / (\text{lb}_m \cdot ^\circ\text{R})$$

$$E = 30000 \text{ Btu} / (\text{lb} \cdot \text{mol})$$

$$R = 1.99 \text{ Btu} / (\text{lb} \cdot \text{mol} \cdot ^\circ\text{R})$$

$$T^{\text{ref}} = 600 \text{ } ^\circ\text{R}$$

$$V_J = 3.85 \text{ ft}^3$$

$$\Delta H = -30000 \text{ Btu} / \text{lb} \cdot \text{mol}$$

$$\rho_J = 62.3 \text{ lb}_m / \text{ft}^3$$

●● 参数的非 SI 单位换算参见[例 3-27]

数学模型

总连续性方程
$$\frac{dV}{dt} = F_0 - F \quad (3)$$

组分 A 的物料平衡方程
$$\frac{d(V C_A)}{dt} = F_0 C_{A0} - F C_A - V k C_A \quad (4)$$

反应器中物料的热量平衡
$$\frac{d(VT)}{dt} = F_0 T_0 - F T - \frac{(-\Delta H) V k C_A}{\rho c_p} - \frac{Q}{\rho c_p} \quad (5)$$

式中
$$Q = \frac{U A_H}{\rho C_p} (T - T_J) \quad (6)$$

反应速率常数
$$k = k_0 \exp(-E/RT) \quad (7)$$

夹套中冷却流体的热量平衡:
$$\frac{d(T_J)}{dt} = \frac{F_J(T_{J0} - T_J)}{V_J} + \frac{Q}{\rho_J V_J c_{pJ}} \quad (8)$$

程序说明 用 ode23s() 计算 (以稳态值作为初值), 其中 Equations() 定义由式(3)~(5)和(8)构成的常微分方程组。若用 ode45(), 则发现 TJ~t 图形有些异样 (最后一个 TJ 值突然变小)。

程序清单 NonIsothermCSTR.m

```
function NonIsothermCSTR
clear all; clc
global F0 F CA0 Kc T0 k0 dH rho rhoJ Cp VJ CJ U Ah FJ TJ0
CA = 0.245; T = 600; TJ = 594.6; V = 48; VC = V*CA; VT = V*T; % 初始条件
TJ0 = 530; F0 = 40; T0 = 530; CA0 = 0.5; Kc = 4; delta = 0.01; % 参数值
CA0 = 0.55; % 浓度扰动
VJ = 3.85; E = 30000; U = 150; Cp = 0.75; rho = 50; k0 = 7.08e10; R = 1.99;
Ah = 250; dH = -30000; CJ = 1.0; rhoJ = 62.3; Tset = 600;
[t, y] = ode23s(@Equations, [0:0.2:4], [V VC VT TJ]); % 求解 ODEs
V = y(:, 1); CA = y(:, 2)/V; T = y(:, 3)/V; TJ = y(:, 4);
FJ = 49.9-Kc*(600-T); F = 40-10*(48-V); Q = U*Ah*(T-TJ)

% 显示图形结果
disp(' 结果:')
disp('          t          CA          T          V          F          TJ          FJ')
disp([t CA T V F TJ FJ])
plot(t, CA), xlabel('t, h'), ylabel('C_A, lbmol/ft^3'), axis([0 4 0.22 0.26])
figure, plot(t, T), xlabel('t, h'), ylabel('T, ^oR'), axis([0 4 600 610])
figure, plot(t, V), xlabel('t, h'), ylabel('V, ft^3'), axis([0 4 40 50])
figure, plot(t, F), xlabel('t, h'), ylabel('F, ft^3/h'), axis([0 4 20 45])
figure, plot(t, TJ), xlabel('t, h'), ylabel('T_J, ^oR'), axis([0 4 590 600])
figure, plot(t, FJ), xlabel('t, h'), ylabel('F_J, ft^3/h'), axis([0 4 20 80])

% -----
function dydt = Equations(t, y)
```

```
global F0 F CA0 Kc T0 k0 dH rho rhoJ Cp VJ CJ U Ah FJ TJ0
```

```
V = y(1); VC = y(2); VT = y(3); TJ = y(4); CA = VC/V; T = VT/V;
FJ = 49.9 - Kc*(600 - T); F = 40 - 10*(48 - V); % feedback controllers
k = k0*exp(-30000/(1.99*T)); % reaction rate
Q = U*Ah*(T - TJ);
```

```
% simultaneous differential equations
```

```
dVdt = F0 - F;
```

```
dVCdt = F0*CA0 - F*CA - V*k*CA;
```

```
dVTdt = F0*T0 - F*T - (dH*V*k*CA+Q)/(rho*Cp); % dH:反应热
```

```
dTJdt = FJ*(TJ0-TJ)/VJ + Q/(rhoJ*VJ*CJ);
```

```
dydt = [dVdt; dVCdt; dVTdt; dTJdt];
```

计算结果 部分结果如图 4-31 所示。

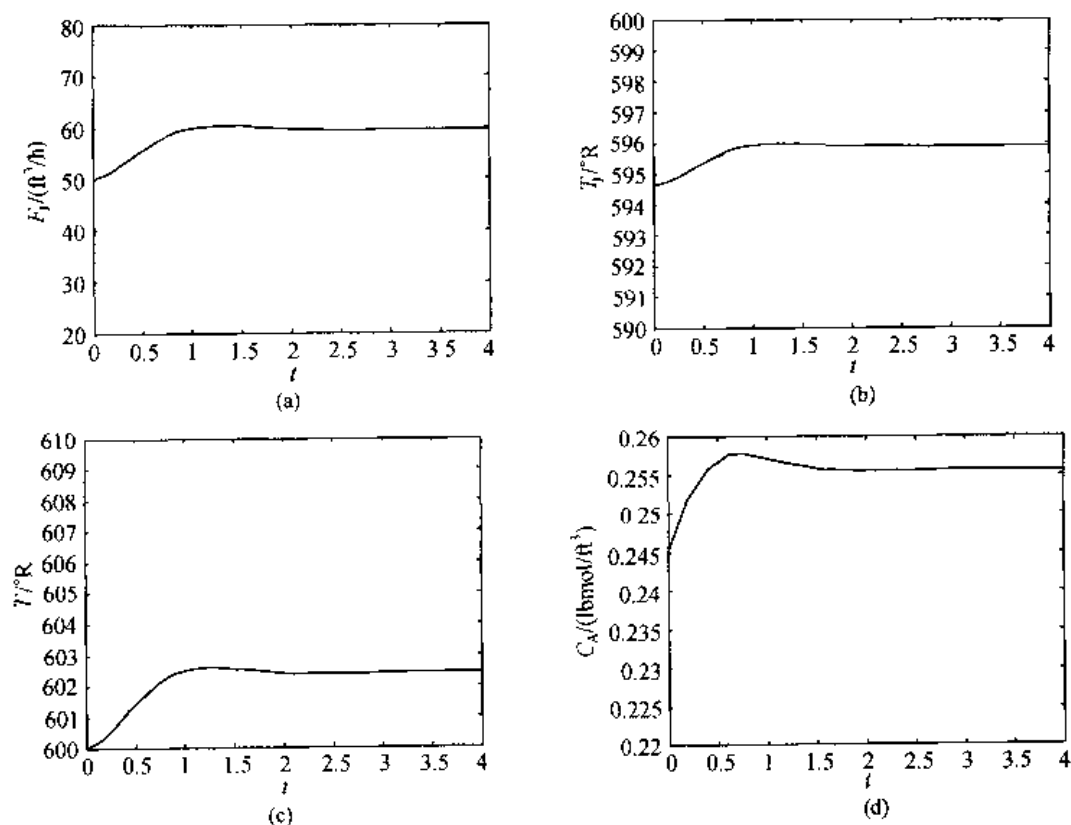


图 4-31 非等温反应器: +10% ΔC_{A0}

符号说明

A_H	传热面积, ft^2	t	时间, h
C_A	A 的浓度, lbmol/ft^3	T	反应温度, $^{\circ}\text{R}$
c_p	比热容, $\text{Btu}/(\text{lb}_m^{\circ}\text{R})$	T_J	夹套温度, $^{\circ}\text{R}$
E	活化能, $\text{Btu}/(\text{lbmol})$	T^{set}	反应温度设定值, $^{\circ}\text{R}$
F	体积流量, ft^3/h	U	总传热系数, $\text{Btu}/(\text{h ft}^2^{\circ}\text{R})$
k	反应速度常数, h^{-1}	V	反应器中的液体容积, ft^3

k_0	指前因子, h^{-1}	V_j	夹套容积, ft^3
K_c	控制增益, $(\text{ft}^3/\text{h})/^\circ\text{R}$	ΔH	反应热, $\text{Btu}/\text{lb}\cdot\text{mol}$
Q	传热量, Btu	ρ	物料密度, $\text{lb}_\text{m}/\text{ft}^3$
R	理想气体常数, $\text{Btu}/(\text{lb}\cdot\text{mol}\cdot^\circ\text{R})$	ρ_j	夹套内流体密度, $\text{lb}_\text{m}/\text{ft}^3$
下标			
0	进料	J	夹套中的冷却介质 (水)

【例 4-24】三个串联 CSTR 等温反应器的闭环控制系统^[2]

前面例 4-3 属于开环 (open loop) 系统, 它不使用反馈控制。本例在例 4-3 的基础上, 增加反馈控制器构成闭环控制系统, 如图 4-32 所示。这里使用一个比例积分控制器, 用于控制离开第三个反应器的产物所含 A 的浓度 C_{A3} , 即通过调节进入第一个反应器的进料浓度 C_{A0} , 使得 C_{A3} 保持在设定值 $C_{A3, \text{set}}$ 附近。若在 $t=0$ 时刻, 给定一个浓度扰动阶跃变化: $C_{AD} = 0.2$, 试模拟该系统的动态行为。

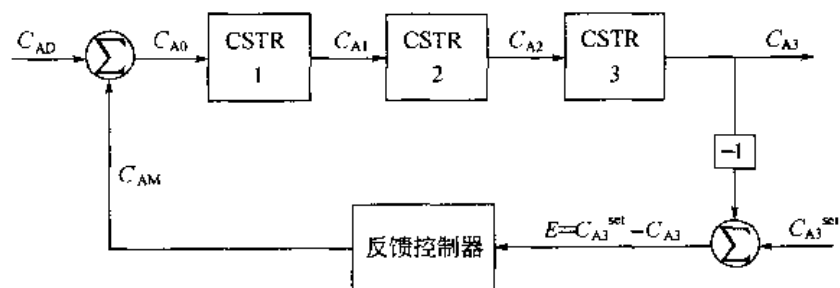


图 4-32 三个串联 CSTR 等温反应器的闭环控制系统

数学模型

反应系统模型 (三个 ODE 方程) 与上例相同, 只是方程中的 C_{A0} 不同, 它与控制方程相关联。所以下面需要导出有关控制方程。

以变量 C_{AD} 表示扰动浓度, C_{AM} 表示由控制器改变的操作浓度, 则

$$C_{A0} = C_{AM} + C_{AD} \quad (1)$$

反馈控制器具有比例作用和积分作用, 它根据 C_{A3} 与设定值 $C_{A3, \text{set}}$ 之间的残差改变 C_{AM} :

$$C_{AM} = 0.8 + K_c \left(E + \frac{1}{\tau_i} \int E dt \right) \quad (2)$$

式中, 0.8 为控制器的偏移修正值, 即 $t=0$ 时 C_{AM} 的值。残差为

$$E = C_{A3, \text{set}} - C_{A3} \quad (3)$$

$$\text{定义 } E_1 = \int E dt, \text{ 则有 } \frac{dE_1}{dt} = E \quad (4)$$

这样, 我们从控制器得到一个附加的 ODE 方程 (4), 将此方程与上例中描述的三个 ODE 方程一起联立求解。

程序说明 用 ode45() 进行数值积分, 积分步长自动调节。比例积分控制器参数取值: $K_c = 30$, $\tau_i = 5 \text{ min}$ 。有关比例积分控制器参数 K_c 和 τ_i 的选取, 可参阅 Luyben^[2]。

程序清单 CSTRs_Closedloop.m

```
function CSTRs_Closedloop    % 三个串联 CSTR 等温反应器的比例反馈闭环控制系统
clear all; clc
```

```

global k Kc CA0 CAD tau CA3set tau1
CA = 1.8; CA1 = 0.4; CA2 = 0.2; CA3 = 0.1; % 初始条件, kmol/m3
CA3set = 0.1; EI = 0; CAD = 0.2; % CAD: disturbance
k = 0.5; tau = 2; Kc = 30; tau1 = 5; stoptime = 8.5; % 参数: k -- min-1, stoptime -- min
[t, y] = ode45(@Equations, [0:0.5:stoptime], [CA1 CA2 CA3 EI]); % Solve ODEs

% Output results
CA1 = y(:, 1); CA2 = y(:, 2); CA3 = y(:, 3); EI = y(:, 4);
E = CA3set - CA3; CAM = 0.8 + Kc*(E + EI/tau1);

disp(' Results:')
disp('      t      CA1      CA2      CA3      CAM'), disp([t y(:,1:3) CAM])
plot(t, CA1, 'b--', t, CA2, 'm--', t, CA3, 'r-', t, CAM, 'k:')
axis([0 t(end)+0.5 min(CA3)-0.01 max(CAM)])
xlabel('Time (min)'), ylabel('C_A_1, C_A_2, C_A_3, C_A_M (kmol/m^3)')
legend('C_A_1','C_A_2','C_A_3','C_A_M')

% -----
function dydt = Equations(t, y)
global k Kc CA0 CAD tau CA3set tau1
CA1 = y(1); CA2 = y(2); CA3 = y(3); EI = y(4);

% feedback controller
E = CA3set - CA3; CAM = 0.8 + Kc*(E + EI/tau1); CA0 = CAM + CAD;

% Model equations
dCA1dt = (CA0-CA1)/tau - k*CA1; dCA2dt = (CA1-CA2)/tau - k*CA2;
dCA3dt = (CA2-CA3)/tau - k*CA3; dEIdt = E;

dydt = [dCA1dt; dCA2dt; dCA3dt; dEIdt];

```

计算结果 当 $t = 0$ 时, 若扰动浓度发生阶跃变化 ($C_{AD} = 0.2$), 则系统动态响应如图 4-33 所示。(相应动态数据可通过运行程序直接观察)。

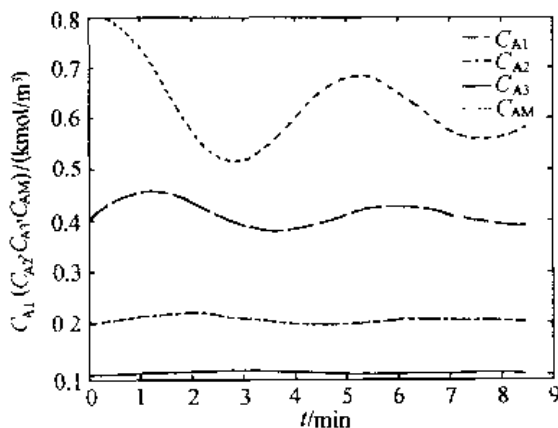


图 4-33 系统动态响应

符号说明

C_{AD}	扰动浓度, kmol/m^3	E	残差 $E = C_{A3, \text{set}} - C_{A3}$, kmol/m^3
C_{AM}	由控制器改变的操作浓度, kmol/m^3	K_C	反馈控制器增益
$C_{A3, \text{set}}$	设定值, kmol/m^3	τ_i	反馈控制器的积分时间常数, min

【例 4-25】 双组分间歇精馏的回流比控制^[1, 2]

在一间歇精馏塔中进行双组分液体混合物的分离, 该塔有 7 块理论板、一个塔顶冷凝器和一个再沸器。通过比例控制回流比 R 可以控制所需要的塔顶出口馏液组成。已知相对挥发度 $\alpha = 3.5$, 塔顶冷凝器中的摩尔滞液量 $M_0 = 100$, 塔板上的摩尔滞液量 $M = 5 \text{ kmol}$, 蒸汽流量 $V = 10 \text{ kmol/h}$, 蒸馏釜处理量为 2500 kmol , 塔釜组成 (摩尔分数) $x_B = 0.8$ 。控制参数: 塔顶馏分组成设定值 $x_{0\text{set}} = 0.9$, 控制增益 $K_C = 100$ 。开车时, 进行全回流操作, 一旦塔顶馏分的组成超过设定值, 即采用比例控制器 $R = K_C(x_{1\text{set}} - x_1)$ 调节回流比 R 。

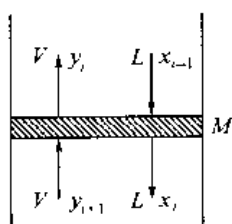


图 4-34 气液两相通过第 i 块板的流动

数学模型 设塔板总数为 N (包括塔顶冷凝器和塔釜再沸器, 本例 $N = 9$), 塔板序号 i 自顶而下排序, 即塔顶冷凝器为 $i = 1$, 塔内塔板为 $i = 2, 3, \dots, N-1$, 塔釜再沸器为 $i = N$ 。假设恒摩尔流、持液量为常数, 下面基于易挥发组分进行物料衡算。

$$\text{对塔顶冷凝器,} \quad M_1 \frac{dx_1}{dt} = Vy_2 - (L + D)x_1 \quad (1)$$

对第 i 块塔板上的易挥发组分 (如图 4-34) 有,

$$M \frac{dx_i}{dt} = L(x_{i-1} - x_i) + V(y_{i+1} - y_i) \quad i = 2, 3, \dots, N-1 \quad (2)$$

对塔釜 (column reboiler),

$$\frac{dM_B}{dt} = L - V \quad (3)$$

$$\frac{d(M_B x_N)}{dt} = Lx_{N-1} - Vy_N \quad (4)$$

VLE 关系:

$$y_i = \frac{\alpha x_i}{1 + (\alpha - 1)x_i} \quad (5)$$

回流比:

$$R = L_0/D - L/D \quad (6)$$

$$D = \frac{V}{R+1} \quad (\text{或 } V = (R+1)D) \quad (7)$$

$$L = V - D \quad (L = RD) \quad (8)$$

控制方程: 初始开车时,

$$R = R_{\text{init}} \quad (\text{全回流操作}) \quad (9)$$

对于塔顶产品馏分,

$$R = K_C(x_{1\text{set}} - x_1) \quad (10)$$

程序说明 开车时, 进行全回流操作, 即 $R = R_{\text{init}}$, 其中 R_{init} 可设置为很大的任意一个值, 如 $R_{\text{init}} = 1 \times 10^{10}$ 。一旦塔顶馏分的组成超过设定值 x_0^{set} , 控制器即根据比例控制关系 $R = K_C(x_{1\text{set}} - x_1)$ 调节回流比 R 。逻辑变量 flag 为一标识符, 当全回流时, flag 为 0; 当处在被控制状态时, flag 为 1。程序用 ode45() 求解由 DistilEqs() 定义的动态 ODE 方程组 (1) ~ (4), 其中控制方程放在函数 DistilEqs() 内于 ODE 方程组 (1) ~ (4) 之前进行计算。

程序清单 见光盘中的 BatchDistill.m。

计算结果 见图 4-35 和图 4-36。

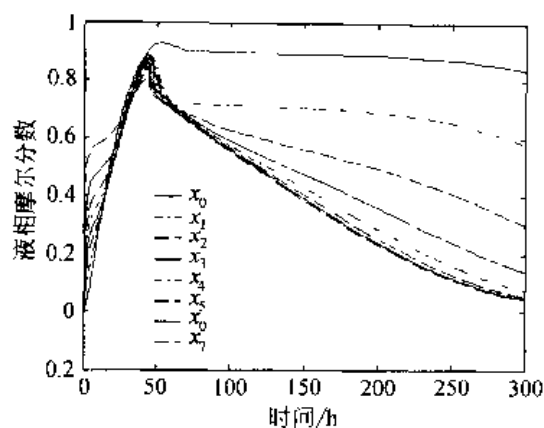


图 4-35 在调节回流比下的组成动态响应

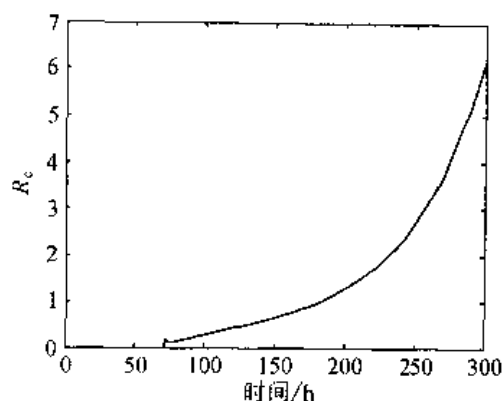


图 4-36 相应被调节的回流比变化

符号说明

D 塔顶馏出液流量, kmol/h
 K_C 控制器增益
 L 液体流量, kmol/h
 M 摩尔滞液量, kmol
 R 回流比

下标

1 塔顶冷凝器
 2~ $N-1$ 塔板
 N 塔板总数
 B 再沸器

V 蒸汽流量, kmol/h
 x 液相摩尔分数
 y 气相摩尔分数
 α 相对挥发度

$init$ 初始条件
 i 第 i 块塔板
 set 控制器设定值

由于篇幅所限, 还有其他一些方面未作介绍, 比如流化床、液-液萃取塔、其他多种生化反应器等, 属于 ODE 问题, 数值模拟方法完全一样。

习 题

4-1 已知反应系统为^[17]: $C_1 \xrightleftharpoons[k_2 C_2 C_3]{k_1 C_1} C_2 \xrightarrow{k_3 C_2^2} C_3$, 描述该系统的微分方程组为:

$$\frac{dC_1}{dt} = -k_1 C_1 + k_2 C_2 C_3$$

$$\frac{dC_2}{dt} = k_1 C_1 - k_2 C_2 C_3 - k_3 C_2^2$$

$$\frac{dC_3}{dt} = k_3 C_2^2$$

$$I.C. \quad C_1(0) = C_0 \quad C_2(0) = C_3(0) = 0$$

已知: $k_1 = 0.04$, $k_2 = 10^4$, $k_3 = 3 \times 10^7$, $C_0 = 1.0$ 。试求解此方程组, 并标绘各组分浓度变化曲线。

4-2 非等温管式反应器模拟计算^[19]:

在一个有夹套管式反应器中进行丙酮气相裂解反应 (吸热), 反应式为 $\text{CH}_3\text{COCH}_3 \rightarrow \text{CH}_2\text{CO} + \text{H}_2$, 纯丙酮以 $T_0 = 1035 \text{ K}$ 和 $p_0 = 162 \text{ kPa}$ 的条件进入反应器, 换热器出口气体温度恒定为 $T_a = 1150 \text{ K}$, 体积流量 $v_0 = 0.002 \text{ m}^3/\text{s}$, 反应器体积 $V_R = 1 \text{ m}^3$, 总传热系数 $U = 110 \text{ W}/(\text{m}^2 \cdot \text{K})$, 传热面积 $A = 150 \text{ m}^2/\text{m}^3$ 反应器, 其他参数的计算公式如下:

$$\text{反应速率常数 } k = 3.58 \exp \left[34222 \left(\frac{1}{1035} - \frac{1}{T} \right) \right] \text{ s}^{-1},$$

$$\text{反应热 } \Delta H_R = 80770 + 6.8(T - 298) - 5.75 \times 10^{-3}(T^2 - 298^2) - 1.27 \times 10^{-6}(T^3 - 298^3) \text{ J/mol},$$

$$\text{丙酮摩尔热容 } c_{pA} = 26.63 + 0.1830T - 45.86 \times 10^{-6}T^2 \text{ J/(mol}\cdot\text{K)}$$

$$\text{乙烯酮摩尔热容 } c_{pB} = 20.04 + 0.0945T - 30.95 \times 10^{-6}T^2 \text{ J/(mol}\cdot\text{K)}$$

$$\text{甲烷摩尔热容 } c_{pC} = 13.39 + 0.0770T - 18.71 \times 10^{-6}T^2 \text{ J/(mol}\cdot\text{K)}$$

试确定气体沿反应器管长的温度曲线。假定反应器内压力均匀。

4-3 FRIPP 加氢裂化集总动力学模型的计算^[21]

芳烃加氢反应网络（三集总）为 $A \xrightleftharpoons[k_2]{k_1} N \xrightarrow{k_3} P$ ，其中，A、N、P 分别表示芳烃、环烷烃及烷烃集总。其动力学方程为： $dC_A/dt = -k_1C_A + k_2C_N$ ， $dC_N/dt = k_1C_A - (k_2 + k_3)C_N$ ， $dC_P/dt = k_3C_N$ ，式中 $k_1 = k_{10}e^{-E_1/RT} (p_{H_2}/6.5)^{\alpha_1}$ ， $k_2 = k_{20}e^{-E_2/RT} (p_{H_2}/6.5)^{\alpha_2}$ ， $k_3 = k_{30}e^{-E_3/RT} (p_{H_2}/6.5)^{\alpha_3}$ 。动力学参数如下表：

$k_{10}/\times 10^{-4}$	$k_{20}/\times 10^{-13}$	$k_{30}/\times 10^{-19}$	$E_1/\text{kJ}\cdot\text{mol}^{-1}$	$E_2/\text{kJ}\cdot\text{mol}^{-1}$	$E_3/\text{kJ}\cdot\text{mol}^{-1}$	α_1	α_2	α_3
6.77	3.68	2.37	65.31	182.6	248.9	0.942	-2.37	-0.52

已知 $C_{A0} = 15\%$ ， $C_{N0} = 30\%$ ， $C_{P0} = 0$ ， $p_{H_2} = 6.5 \text{ MPa}$ ， $T = 390^\circ\text{C}$ 。试计算 1 h 内的浓度动态变化。

4-4 多孔催化剂中伴有复杂反应的扩散问题^[11, 18]

催化裂化柴油（A）生成汽油（B），伴随产生轻的气体、焦炭等（C）。反应系统如图 4-37 所示。与 A 有关的两个反应是 2 级，而与 B 有关的反应是 1 级。

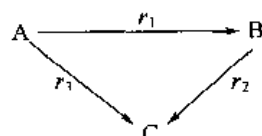


图 4-37 催化裂化的集中参数模型

该反应系统典型的工业催化剂是硅-铝晶格（Matrix）M 中含 5.0%（质量分数）分子筛 Z 的混合物，分子筛催化剂的催化活性比较高，对汽油产品的选择性也较晶格 M 的高，但分子筛的有效扩散系数比较低。下表是此催化反应系统的典型数据（Martin, 1987）。

在特定的应用中，催化剂颗粒可近似看成厚度为 L 的薄片，其动力学和传递特性可看成分子筛 Z 和晶格 M 的加权平均值。

$$D_{Ax} = uD_{AZ} + (1-u)D_{AM} \quad (a)$$

$$D_{Bx} = uD_{BZ} + (1-u)D_{BM} \quad (b)$$

$$k_x = uk_{xz} + (1-u)k_{xm} \quad (i=1, 2, 3) \quad (c)$$

式中， u 为分子筛的局部质量分数。注意到分子筛的总质量分数总是 5%，在催化剂颗粒中的任意位置上不能超过 20%。因此，

$$\frac{1}{L} \int_0^L u dx = 0.05 \quad (u \leq 0.05) \quad (d)$$

边界条件

$$\text{在颗粒中心处通量为 0: } x=0, \quad \frac{dC_A}{dx} = \frac{dC_B}{dx} = 0 \quad (e)$$

$$\text{在颗粒表面处: } x=L, \quad C_A = 1.6 \times 10^{-5} \text{ mol/cm}^3, \quad C_B = C_C = 0 \quad (f)$$

B 的选择性是 x 方向 B 的质量通量除以反方向 A 的质量通量，即

$$x=L, \quad S_B = \frac{-D_{Bx}(dC_B/dx)}{D_{Ax}(dC_A/dx)} \quad (g)$$

$$\text{对此伴有反应的扩散过程，微分方程为: } D_{Ax} \frac{d^2 C_A}{dx^2} = k_{3x} C_A^2 + k_{1x} C_A^2 \quad (h)$$

$$D_{Bx} \frac{d^2 C_B}{dx^2} = k_{2x} C_B - k_{1x} C_A^2 \quad (i)$$

式中参数见表 4-4。

表 4-4 参数值^[6]

$k_{1Z} = 10^8 \text{ cm}^3/(\text{mol}\cdot\text{s})$	$k_{1M} = 10^7 \text{ cm}^3/(\text{mol}\cdot\text{s})$
$k_{2Z} = 8 \times 10^2 \text{ s}^{-1}$	$k_{2M} = 10^2 \text{ s}^{-1}$
$k_{3Z} = 3 \times 10^6 \text{ cm}^3/(\text{mol}\cdot\text{s})$	$k_{3M} = 8 \times 10^6 \text{ cm}^3/(\text{mol}\cdot\text{s})$
$D_{AZ} = 10^{-6} \text{ cm}^2/\text{s}$	$D_{AM} = 10^{-3} \text{ cm}^2/\text{s}$
$D_{BZ} = 10^{-5} \text{ cm}^2/\text{s}$	$D_{BZ} = 10^{-2} \text{ cm}^2/\text{s}$
$L = 0.002 \text{ cm}$	

试求解催化剂颗粒内的浓度分布。

参 考 文 献

- 1 J. Ingham, I. J. Dunn, E. heinzle, J. E. Prenosil, Chemical Engineering Dynamics. 2nd edition. Wiley-VCH, 2000, 293, 376, 551, 556, 545
- 2 Luyben, W. L. Process Medelling, Simulation and Control for Chemical Engineers. McGraw-Hill, 1990. (p.41, 119, 46, 124, 122)
- 3 Smith, J. M., Chemical Engineering Kinetics. 3rd edition. MCGraw-Hill, 1981, 159. 中译本, 王建华、许学书、黄世英、刘栋昌和江礼科译. 化工动力学(第三版). 化学工业出版社和成都科技出版社, 1988
- 4 Froment, G. F. Ind. Eng. Chem., 59, No. 2, 18(1967). 或 Froment, G. F., and Bischoff, K. B. Chemical Reactor Analysis and Design, John Wiley and Sons. New York, 1979. 中译本, 邹仁鉴等译. 反应器分析与设计. 北京: 化学工业出版社, 1985, 515, p.575~579
- 5 Ramirez W. F., Computational Methods for Process Simulation, Butterworth Publishers, 1989, 167~173
- 6 V. G. Jenson and G. V. Jeffeys. Mathematical Methods in Chemical Engineering. 2nd Edition, London: Academic Press Inc., 1977. 中译本, 台德荣译. 化工数学方法. 第二版. 北京: 化学工业出版社, 1982, 394
- 7 Geankoplis, C. J. Transport Processes and Unit Operations. 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1993
- 8 张建侯和许锡恩编著. 化工过程分析与计算机模拟. 北京: 化学工业出版社, 1989, 255
- 9 Mark E. Davis, Numerical Methods and Modeling for Chemical Engineers. John Wiley & Sons, Inc., 1984. (Davis, Chater 2)
- 10 Fogler H. S. Elements of Chemical Reaction Engineering, 3rd Edition. Prentice Hall PTR, Upper Saddle River, New Jersey, 1999, 733~734, 182~187
- 11 M. B. Cutlip and M. Shacham, Problem Solving in Chemical Engineering, 1999, 282, 390
- 12 黄华江, 戴迎春, 袁渭康. 连续回热型气-固化学热泵制冷系统的操作优化. 化学反应工程与工艺, 2003(2).
- 13 Hua-Jiang Huang, Guo-Bin Wu, Jiao Yang, Ying-Chun Dai*, Wei-Kang Yuan, and Hui-Bo Lu. Modeling of Gas-solid Chemisorption in Chemical Heat Pumps. SEPERATION AND PURIFICATION TECHNOLOGY. 2003. (in press)
- 14 Raman R., Chemical Process Computation. London: Elsevier Applied Science Publishers, 1985. 中译本, 许锡恩, 张福芝, 王保国, 吴诗华译. 化工过程计算. 北京: 化学工业出版社, 1992
- 15 E. Bruce Nauman. Chemical Reactor Design, Optimization, and Scaleup. McGraw-Hill Inc., 2001, 453~455
- 16 Bequette, B. Wayne, Process Dynamics: Modelling, Analysis, and Simulation, Prentice-Hall PTR, A Simon & Schuster Company, New Jersey, 1998.
- 17 Finlayson Bruce A. Nonlinear Analysis in Chemical Engineering. New York: McGraw-Hill, 1980, 2
- 18 Martin, G. R., White, C. W., and Dadyburjor, D. B., "Design of Zeolite/Silica-Alumina Catalysts for Triangular Cracking Reactions", Journal of Catalysis, 106, 116~124 (1987).
- 19 A. Constantinides and N. Mostoufi, Numerical Methods for Chemical Engineers with MATLAB Applications, Prentice Hall PRT, Upper Saddle River, New Jersey, 1999.
- 20 Gordon L. Amidon, Ping I. Lee and Elizabeth M. Topp ed. Transport Processes in Pharmaceutical Systems. Marcel Dekker Inc., New York, 2000
- 21 韩崇仁主编. 加氢裂化工艺与工程. 北京: 中国石化出版社, 2001, 631~633

第5章 化工中的偏微分方程及其求解

偏微分方程(Partial Differential Equation, 简称PDE)就是涉及两个自变量以上的微分方程。由于PDE方程能够定量描述多变量的物理现象,因而它在科学研究与工程技术领域得到广泛应用。尤其在化学工程领域,为了更好地进行过程设计、优化和控制,经常需要了解化工设备(如反应器)中温度、浓度和速度在不同空间上的分布以及随时间的动态变化规律,因而涉及到许多偏微分方程的问题。

一般来说,偏微分方程很难得到分析解,通常利用数值方法求出其数值解。为此,本章主要介绍三种常用的偏微分方程数值近似解法:有限差分法(finite-difference methods, FDM)、正交配置法(collocation methods, CM)和有限元法(finite-element methods, FEM)。重点介绍基于MATLAB的有限元求解PDE方法。

5.1 概述

5.1.1 偏微分方程的常见类型

$$A \frac{\partial^2 u}{\partial x^2} + B \frac{\partial^2 u}{\partial x \partial y} + C \frac{\partial^2 u}{\partial y^2} + D \frac{\partial u}{\partial x} + E \frac{\partial u}{\partial y} + Fu = f(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}) \quad (5-1)$$

式中, A , B 和 C 为常数时,称为拟线性(quasilinear)偏微分方程。偏微分方程可分为三种类型:

当 $B^2 - 4AC < 0$ 时,方程为椭圆型(elliptic)偏微分方程;

当 $B^2 - 4AC = 0$ 时,方程为抛物型(parabolic)偏微分方程;

当 $B^2 - 4AC > 0$ 时,方程为双曲型(hyperbolic)偏微分方程。

最经典的三种偏微分方程是波动方程、导热方程和拉普拉斯方程,下面做简要介绍。

(1) 导热方程(抛物型): 一维动态线性热传导方程

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \quad (5-2)$$

(2) 拉普拉斯方程(椭圆型): 如稳态静电场和稳态温度分布模型

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad (5-3)$$

(3) 波动方程(双曲型): 一维弦震动模型

$$\frac{\partial^2 u}{\partial t^2} = \alpha^2 \frac{\partial^2 u}{\partial x^2} \quad (5-4)$$

在化学工程中,最普遍的PDE方程是描述有化学反应或无化学反应的静止流体或流体流动的传热和传质方程。PDE方程主要包括初值型PDE方程和边值型PDE方程,初值问题就是至少一个自变量有一个开域的问题;而在边值问题中全部自变量都有闭域。在化学工程中时间是常见的开域变量,而描述有限空间的坐标是最常见的闭域变量。因此,PDE方程主要是以边值问题的形式出现的椭圆型方程和以初值问题的形式出现的抛物型方程,而双曲型方

程则很少见。

5.1.2 偏微分方程的边界条件

边界条件一般可分为以下四种类型:

(1) Dirichlet 边界条件 (又称第一类边界条件) — 已知函数在边界 $\partial\Omega$ 上给定函数 $u(x, y)$ 的值, 即 $u|_{\partial\Omega} = \varphi(x, y)$ (5-5)

(2) Neumann 边界条件 (又称第二类边界条件) — 已知导数在边界 $\partial\Omega$ 上给定函数 $u(x, y)$ 的外法向导数值, 即 $\frac{\partial u}{\partial n}|_{\partial\Omega} = \varphi(x, y)$ (5-6)

其中, n 为边界 $\partial\Omega$ 的外法向。

(3) Robin 边界条件 (混合边界条件, 又称第三类边界条件) 在边界 $\partial\Omega$ 上给定函数 $u(x, y)$ 与其外法向导数的线性组合的值, 即

$$\left(\frac{\partial u}{\partial n} + qu \right)|_{\partial\Omega} = \varphi(x, y) \quad (5-7)$$

其中, $q = q(x, y)$ 为已知函数。

(4) 积分-微分边界条件

在传质研究中, 当物质穿过界面进入或离开一个限定的体积而造成推动力变化时, 经常出现这类边界条件。具体从略, 请参见文献^[1]。

在以上的四种边界条件中, 前三种最为常见。

5.1.3 偏微分方程数值方法简介

(1) 有限差分法

有限差分法在求解域内画了很多横竖交叉的规则网格, 交叉点称为网格点 (grid points), 在所有的网格点上, PDE 方程的导数用有限差分近似代替, 最后得到代数方程组 (若是线性 PDE 方程, 则得到线性代数方程组)。有限差分法一般具有较好的数值稳定性, 是求解微分方程最基本的数值方法。该方法的缺点是: 不能求出网格点之间的解; 很难用于不规则几何形状或者像导热系数突变的情况。

(2) 正交配置法

正交配置法是加权余量法 (Method of Weighted Residuals) 的一种形式, 它是求解常微分方程边值问题和偏微分方程边值问题的一种有效方法, 在化学工程计算中已得到广泛应用。

加权余量法的基本思想是: 首先选择一多项式作为试函数, 将此试函数代入微分方程, 再求出多项式的根作为配置点, 令在各配置点试函数代入微分方程后的残差 (“余量”) 为零, 得到关于多项式系数的代数方程组, 然后求解此方程组得到多项式中各项的系数, 由此得到的多项式即为微分方程的近似解析解。

(3) MOL 法

线上法 (Method of Lines, MOL): 是将一个自变量当成连续变量, 而对其余的自变量用有限差分法或者正交配置法进行离散, 从而把偏微分方程转变为常微分方程组, 然后用龙格-库塔法积分求解。该方法可用于求解一维动态和二维稳态 PDE 方程。

(4) 有限元法

有限元法首先把求解域划分为大量的单元 (elements), 其中任意大小和方向的三角形网格尤其适用于二维的情况。三角形顶点称为节点 (nodes), 并与相邻单元相连接。最简单的情况, 就是在第 i 个三角形内因变量以公式 $u_i = a_i + b_i x_i + c_i y_i$ 表示, 式中常数 a_i, b_i 和 c_i ($i = 1,$

2,...,n) 未知。考虑边界条件，最后也得到 n 个代数方程组（若是线性 PDE 方程，也得到线性代数方程组）。有限元法的优点是：容易处理复杂几何区域，容易与各种有用的边界条件结合使用，容易处理不规则几何形状或者像导热系数突变的情况；还可提供整个求解域的连续解，而不仅仅是在节点上的解。

MATLAB 偏微分方程(PDE)工具箱就是采用有限元法求解 PDE 问题的。

5.2 用有限差分法解偏微分方程

5.2.1 有限差分法

有限差分法是偏微分方程离散化最基本的方法，常见的差分表达式如表 5-1 所示。

表 5-1 几种常见的差分表达式

导数	差分表达式		截断误差
$\frac{\partial u}{\partial x}\bigg _{i,j,k}$	向前	$\frac{1}{\Delta x}(u_{i+1,j}^k - u_{i,j}^k)$	$o(\Delta x)$
	向后	$\frac{1}{\Delta x}(u_{i,j}^k - u_{i-1,j}^k)$	$o(\Delta x)$
	中心	$\frac{1}{2\Delta x}(u_{i+1,j}^k - u_{i-1,j}^k)$	$o(\Delta x^2)$
$\frac{\partial^2 u}{\partial x^2}\bigg _{i,j,k}$	向前	$\frac{1}{\Delta x^2}(u_{i+2,j}^k - 2u_{i+1,j}^k + u_{i,j}^k)$	$o(\Delta x)$
	向后	$\frac{1}{\Delta x^2}(u_{i,j}^k - 2u_{i-1,j}^k + u_{i-2,j}^k)$	$o(\Delta x)$
	中心	$\frac{1}{\Delta x^2}(u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k)$	$o(\Delta x^2)$
$\frac{\partial^2 u}{\partial x \partial y}\bigg _{i,j,k}$	向前	$\frac{1}{\Delta x \Delta y}(u_{i+1,j+1}^k - u_{i,j+1}^k - u_{i+1,j}^k + u_{i,j}^k)$	$o(\Delta x + \Delta y)$
	向后	$\frac{1}{\Delta x \Delta y}(u_{i,j}^k - u_{i,j-1}^k - u_{i-1,j}^k + u_{i-1,j-1}^k)$	$o(\Delta x + \Delta y)$
	中心	$\frac{1}{4\Delta x \Delta y}(u_{i+1,j+1}^k - u_{i-1,j+1}^k - u_{i+1,j-1}^k + u_{i-1,j-1}^k)$	$o(\Delta x^2 + \Delta y^2)$

表 5-1 中只列出在 x 方向上的差分， y 方向及非稳态项 $\partial u / \partial t$ ，也有向前、向后和中心差分等几种格式，其表达式与表 5-1 所列类似。对非稳态项 $\partial u / \partial t$ 项，只需维持下标不变，而使上标作相应的变化，且空间步长改为时间步长 Δt 即可。如 $\partial u / \partial t$ 的中心差分式为

$$\frac{\partial u}{\partial t} = \frac{1}{2\Delta t}(u_{i,j}^{k+1} - u_{i,j}^{k-1})$$

5.2.2 一维动态 PDE 模型的求解

有限差分法主要包括显式差分法和隐式差分法（如 Crank-Nicholson 差分格式）两种。下面举例说明。

1. 显式差分法

【例 5-1】 用显式差分法求解一维热传导方程（抛物型 PDE 方程）：

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{1a}$$

$$\text{I.C.} \quad \text{对 } t = 0 \text{ 和 } 0 \leq x \leq 10, \quad T = 0 \text{ }^\circ\text{C} \tag{1b}$$

$$\text{B.C.} \quad \text{对 } x = 0 \text{ cm 和所有 } t, \quad T = 100 \text{ }^\circ\text{C} \tag{1c}$$

对 $x=10\text{ cm}$ 和所有 t , $T=0^\circ\text{C}$ (1d)

热扩散系数 $\alpha = 2.0\text{ m/s}$ 。

解 考虑矩形域 $R = \{(x, t): 0 \leq x \leq a, 0 \leq t \leq b\}$, 网格划分为 $(n-1) \times (m-1)$, 步长 $\Delta x = h$ 和 $\Delta t = \tau$, 如图 5-1 所示。从最底行 $t = t_1 = 0$ (即 $j = 1$) 开始计算, 这时 $u(x_i, t_1) = f(x_i)$, 求解各个网格节点上 u 的近似解, 即 $\{u(x_i, t_j): i = 1, 2, \dots, n\}$, $j = 2, 3, \dots, m$ 。

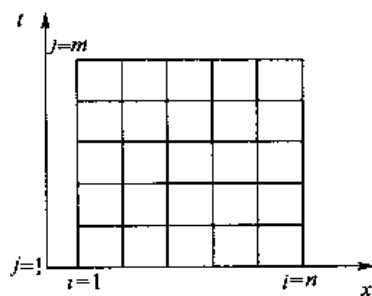


图 5-1

在节点 (x_i, t_j) 处, 对 $\frac{\partial u}{\partial t}$ 用向前差分公式, $\frac{\partial^2 u}{\partial x^2}$ 用中心差分公式, 方程 (1a) 变为

$$\frac{u_{i,j+1} - u_{i,j}}{\tau} = \alpha \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

$$u_{i,j+1} = (1 - 2r)u_{i,j} + r(u_{i+1,j} + u_{i-1,j}) \quad (1e)$$

即

$$\text{式中 } r = \frac{\alpha\tau}{h^2}。$$

(程序从略)

2. 隐式差分法——Crank-Nicolson 差分格式

【例 5-2】 用 Crank-Nicolson 差分格式求解[例 5-1]的模型方程。

解 ① 用 Crank-Nicolson 差分格式将 PDE 问题离散化

在节点 $(x_i, t_{j+1/2})$ 处 (见图 5-2), 对 $\frac{\partial T}{\partial t}$ 和 $\frac{\partial^2 T}{\partial x^2}$ 均用相应的中心差分公式代替, 即

$$\left. \frac{\partial T}{\partial t} \right|_{i,j+\frac{\tau}{2}} = \frac{T_{i,j+1} - T_{i,j}}{2(\tau/2)}$$

$$\left. \frac{\partial^2 T}{\partial x^2} \right|_{i,j+\frac{\tau}{2}} = \left(\left. \frac{\partial^2 T}{\partial x^2} \right|_{i,j} + \left. \frac{\partial^2 T}{\partial x^2} \right|_{i,j+1} \right) / 2 = \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{2h^2}$$

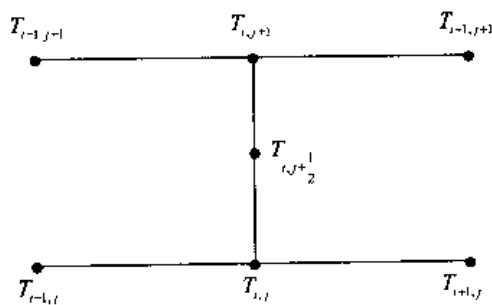


图 5-2 Crank-Nicolson 法的节点计算

将这两个中心差分公式代入例 5-1 的模型方程得 Crank-Nicolson 差分格式

$$-rT_{i-1,j+1} + (2 + 2r)T_{i,j+1} - rT_{i+1,j+1} = rT_{i-1,j} + (2 - 2r)T_{i,j} + rT_{i+1,j}, \quad i = 2, 3, \dots, n-1 \quad (1)$$

$$\text{式中 } r = \frac{\alpha\tau}{h^2}。$$

第一个方程和最后一个方程使用边界条件, 即:

$$T_{1,j} = T_{1,j+1} = c_1 \quad (2)$$

$$T_{n,j} = T_{n,j-1} = c_2 \quad (3)$$

所以方程 (1) 可写成三对角矩阵 $AX = B$, 即

$$\begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 & 4 \end{bmatrix} \begin{bmatrix} T_{2,j+1} \\ T_{3,j+1} \\ \vdots \\ T_{p,j+1} \\ T_{n-2,j+1} \\ T_{n-1,j+1} \end{bmatrix} = \begin{bmatrix} 2c_1 + T_{3,j} \\ T_{2,j} + T_{4,j} \\ \vdots \\ T_{p-1,j} + T_{p+1,j} \\ T_{n-3,j} + T_{n-1,j} \\ T_{n-2,j} + 2c_2 \end{bmatrix} \quad (4)$$

② 程序说明

原 PDE 问题的通用格式为: $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$ (5a)

初始条件 ($t=0$ 和 $0 \leq x \leq a$): $u(x,0) = f(x)$ (5b)

边界条件 ($x=0$ 和 $0 \leq t$): $u(0,t) = g_1(t) = c_1$ (5c)

$u(a,t) = g_2(t) = c_2$ (5d)

程序按此通用格式编写, 其中 $a=10$, $b=8$, $c_1=100$, $c_2=0$, $\alpha=2.0$ 。另外, 程序取节点数为 $n=6$ 和 $m=8$, 且只计算 8 秒钟内的动态变化规律 ($b=8$)。

主程序为 PDE1Dd_CrankNicolson.m, 它调用两个外部函数 CrankNicolson.m 和 TDMA.m (三对角阵算法)。

③ 程序清单:

PDE1Dd_CrankNicolson.m:

```
function PDE1Dd_CrankNicolson
clear all; clc
c1 = 100; c2 = 0; a = 10; b = 8; alpha = 2; n = 6; m = 8;
U = CrankNicolson(@ic, c1, c2, a, b, alpha, n, m)
% -----
function f = ic(x)
f = 0;
```

CrankNicolson.m:

```
function U = CrankNicolson(f, c1, c2, a, b, alpha, n, m)
% Initialize parameters and U
h = a/(n-1); k = b/(m-1); r = alpha*k/h^2; s1 = 2+2/r; s2 = 2/r-2; U = zeros(n, m);
% Boundary conditions
U(1, 1:m) = c1; U(n, 1:m) = c2;
% Generate first row (that is, the first column in matrix U)
U(2:n-1, 1) = feval(f, h:h:(n-2)*h);
```



```

% Form the diagonal and off-diagonal elements of A and
% the constant vector B and solve tridiagonal system AX=B
Vd(1,1:n) = s1*ones(1, n);  Vd(1) = 1;  Vd(n) = 1;
Va = -ones(1, n-1);  Va(n-1) = 0;
Vc = -ones(1, n-1);  Vc(1) = 0;
Vb(1) = c1;  Vb(n) = c2;
for j=2:m
    for i=2:n-1
        Vb(i) = U(i-1, j-1) + U(i+1, j-1) + s2*U(i, j-1);
    end
    X = TDMA(Va, Vd, Vc, Vb);
    U(1:n, j) = X';
end
U = U'

```

TDMA.m: 见光盘中的 *TDMA.m*。

④ **计算结果** 节点数取为 $n=6$ 和 $m=8$, 则 $t=8s$ 时的温度分布为:

100.0000 72.1210 47.3272 27.4209 12.2725 0

5.2.3 二维稳态 PDE 方程组的求解

由于反应器常常是传质、传热和反应的耦合, 因此在反应器的模拟计算过程(设计、操作优化)中, 经常出现初值型偏微分方程组的问题。

导数边界条件的处理

第二类和第三类边界条件都含有必须用有限差分表达式代替的导数, 而由于要计算边界上的导数, 故需要边界两边的因变量的点值。有效的解决方法是, 考虑边界外面一个外加“虚设”的点上的温度。为保持对称, 这个虚设点上的温度必须等于边界内对应点上的温度。现在可以按由有限差分方法导出的一般迭代公式计算边界点的温度, 因为边界点是扩充问题的一个内点。下面通过例子加以说明。

【例 5-3】用有限差分法求解固定床反应器二维拟均相稳态模型(二维稳态 PDE 方程组)^[1]

在一管式催化反应器中进行乙苯的催化脱氢反应, 所用原料为乙苯和水蒸气的气体混合物。

$$\text{动力学方程为} \quad r_c = k(p_E - \frac{p_S p_H}{K}) \quad (1)$$

式中, p_E 、 p_S 和 p_H 分别是乙苯、苯乙烯和氢气的分压(bar)^①; r_c 的单位为 $\text{kmol}/(\text{b kg 催化剂})$ 。

$$\text{反应速率常数为} \quad k = 12600 \exp(-11000/T) \quad (2)$$

$$\text{平衡常数为} \quad K = 0.027 \exp[0.021(T-773)] \quad (3)$$

式中, T 表示温度, K。

已知反应管的内径 $2a = 10 \text{ cm}$, 乙苯和水蒸气在 600°C 下分别以 $0.069 \text{ kmol} \cdot \text{h}^{-1}$ 和 $0.69 \text{ kmol} \cdot \text{h}^{-1}$ 的加料速率进入反应器, 相当于总质量速率为 $G = 2500 \text{ kg} \cdot \text{h}^{-1} \cdot \text{m}^{-2}$ 。反应器管外

① $1\text{bar}=10^5\text{Pa}$ 。

用速率为 $F = 130 \text{ kg} \cdot \text{h}^{-1}$ 的烟道气与反应混合物逆流加热反应管, 烟道气出口温度为 620°C 。其他数据为: 催化剂的堆积密度 $\rho = 1440 \text{ kg/m}^3$, 操作压力 $p = 1.2 \text{ bar}$, 乙苯的反应热 $\Delta H = 140000 \text{ kJ} \cdot \text{mol}^{-1}$, 床层有效导热系数 $\lambda_e = 0.45 \text{ W} \cdot \text{m}^{-1} \cdot \text{K}^{-1}$, 有效扩散系数与气流速度之比 $D_e/u = 0.000427 \text{ m}$, 反应混合物比热容 $c_p = 2.18 \text{ kJ} \cdot \text{kg}^{-1} \cdot \text{K}^{-1}$, 烟道气比热容 $C_p' = 1.0 \text{ kJ} \cdot \text{kg}^{-1} \cdot \text{K}^{-1}$ 。试计算反应管轴向和径向的温度分布和转化率分布, 并计算需要多长的反应器管道才能使乙苯转化率达 45%。

解 ① 建立固定床反应器二维拟均相稳态模型

在反应管内取任意体积微元, 分别作热量衡算和物料衡算, 得到如下传热和传质方程:

$$\frac{\partial T}{\partial z} - a_1 \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right) + b_1 r_c = 0 \quad (4)$$

$$\frac{\partial x}{\partial z} - a_2 \left(\frac{\partial^2 x}{\partial r^2} + \frac{1}{r} \frac{\partial x}{\partial r} \right) - b_2 r_c = 0 \quad (5)$$

式中, z 为反应管的轴向距离, m; r 为反应管的径向距离, m。常数 a_1 , b_1 , a_2 和 b_2 如下:

$$a_1 = \frac{\lambda_e}{Gc_p}, \quad b_1 = \frac{\Delta H \rho}{Gc_p}, \quad a_2 = \frac{D_e}{u}, \quad b_2 = \frac{\rho}{u_0 c_0} \quad (6)$$

式中, C_0 为苯的反应管入口处的浓度。

反应速率 r_c 是温度和转化率的非线性函数。按反应计量关系可得到各组分的分压:

$$p_E = \frac{1.2(1-x)}{11+x}, \quad p_S = F_H = \frac{1.2x}{11+x} \quad (7)$$

$$\text{代入 (1), 得} \quad r_c = 15100 \exp(-11000/T) \left[\frac{1-x}{11+x} - \frac{1.2x^2}{K(11+x)^2} \right] \quad (8)$$

$$\text{传热方程 (4) 的初始及边界条件: } z=0, T=T_0 \quad (9)$$

$$r=0, \frac{\partial T}{\partial r} = 0 \quad (10)$$

式中, T_0 为混合原料气在管入口处的温度, $T_0 = 873 \text{ K}$ 。

右边界 (管内壁) 条件, 可通过取长度为 δz 的微元管进行热量衡算导出:

$$\delta z F c_p' \frac{\partial T}{\partial z} = 2\pi a \delta z \lambda_e \frac{\partial T}{\partial r} \Big|_{r=a}$$

$$\text{即} \quad r=a, \quad \frac{\partial T}{\partial z} = C_1 \frac{\partial T}{\partial r} \quad (11)$$

$$\text{式中,} \quad C_1 = \frac{2\pi a \lambda_e}{F c_p'} \quad (12)$$

$$\text{传质方程 (5) 的初始及边界条件: } z=0, x=0 \quad (13)$$

$$r=0, \frac{\partial x}{\partial r} = 0 \quad (14)$$

$$r=a, \frac{\partial x}{\partial r} = 0 \quad (15)$$

如此, 由式 (4)、(5) 和式 (9~11)、式 (13~15) 构成偏微分方程组的定解问题。

② 偏微分方程的离散化

下面使用 Crank-Nicholson 方法对方程 (4) 进行离散化。将网格划分为 $n \times m$ (径向和轴向的网格数分别为 n 和 m), 步长分别为 Δr 和 Δz , 即 $r = i\Delta r, z = j\Delta z, i = 0, 1, \dots, n; j = 0, 1, \dots, m$ 。

$$\begin{aligned} \text{把 } \left. \frac{\partial T}{\partial z} \right|_{i,j+\frac{1}{2}} &= \frac{T_{i,j+1} - T_{i,j}}{2(\Delta z/2)} = \frac{T_{i,j+1} - T_{i,j}}{\Delta z} \\ \left. \frac{\partial^2 T}{\partial r^2} \right|_{i,j+\frac{1}{2}} &= \left(\left. \frac{\partial^2 T}{\partial r^2} \right|_{i,j} + \left. \frac{\partial^2 T}{\partial r^2} \right|_{i,j+1} \right) \div 2 = \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{2\Delta r^2} \\ \left. \frac{\partial T}{\partial r} \right|_{i,j+\frac{1}{2}} &= \left(\left. \frac{\partial T}{\partial r} \right|_{i,j} + \left. \frac{\partial T}{\partial r} \right|_{i,j+1} \right) \div 2 = \frac{T_{i+1,j} - T_{i-1,j} + T_{i+1,j+1} - T_{i-1,j+1}}{4\Delta r} \\ r_{ci} \Big|_{i,j+\frac{1}{2}} &= \frac{r_{ci,j+1} + r_{ci,j}}{2} \end{aligned}$$

$$\begin{aligned} \text{代入式 (4) 得: } \frac{T_{i,j+1} - T_{i,j}}{\Delta z} - a_1 \cdot \left(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{2(\Delta r)^2} \right. \\ \left. + \frac{1}{i\Delta r} \cdot \frac{T_{i+1,j} - T_{i-1,j} + T_{i+1,j+1} - T_{i-1,j+1}}{4\Delta r} \right) + b_1 \cdot \frac{r_{ci,j+1} + r_{ci,j}}{2} = 0 \end{aligned}$$

$$\begin{aligned} \text{化简得: } (1+M)T_{i,j+1} &= \frac{M}{2} \left[\left(1 - \frac{1}{2i} \right) (T_{i-1,j} + T_{i-1,j+1}) + \left(1 + \frac{1}{2i} \right) (T_{i+1,j} + T_{i+1,j+1}) \right] \\ &\quad + (1-M)T_{i,j} - \frac{b_1}{2} \Delta z (r_{ci,j} + r_{ci,j+1}) \end{aligned} \quad (16)$$

$$\text{式中 } M = \frac{a_1 \Delta z}{\Delta r^2} \quad (17)$$

类似地, 方程 (5) 的差分格式为

$$\begin{aligned} (1+M')x_{i,j+1} &= \frac{M'}{2} \left[\left(1 - \frac{1}{2i} \right) (x_{i-1,j} + x_{i-1,j+1}) + \left(1 + \frac{1}{2i} \right) (x_{i+1,j} + x_{i+1,j+1}) \right] \\ &\quad + (1-M')x_{i,j} + \frac{b_2}{2} \Delta z (r_{ci,j} + r_{ci,j+1}) \end{aligned} \quad (18)$$

$$\text{式中 } M' = \frac{a_2 \Delta z}{\Delta r^2} \quad (19)$$

可将式 (16) 和式 (18) 改写为如下迭代公式

$$T_{i,j+1} = F_{i,j} + \frac{1}{1+M} \left\{ \frac{M}{2} \left[\left(1 - \frac{1}{2i} \right) T_{i-1,j+1} + \left(1 + \frac{1}{2i} \right) T_{i+1,j+1} \right] - \frac{b_1}{2} \Delta z r_{ci,j+1} \right\} \quad (20)$$

$$x_{i,j+1} = G_{i,j} + \frac{1}{1+M'} \left\{ \frac{M'}{2} \left[\left(1 - \frac{1}{2i} \right) x_{i-1,j+1} + \left(1 + \frac{1}{2i} \right) x_{i+1,j+1} \right] + \frac{b_2}{2} \Delta z r_{ci,j+1} \right\} \quad (21)$$

式中,

$$F_{i,j} = \frac{1}{1+M} \left\{ \frac{M}{2} \left[\left(1 - \frac{1}{2i} \right) T_{i-1,j} + \left(1 + \frac{1}{2i} \right) T_{i+1,j} \right] + (1-M)T_{i,j} - \frac{b_1}{2} \Delta z r_{ci,j} \right\} \quad (22)$$

$$G_{i,j} = \frac{1}{1+M'} \left\{ \frac{M'}{2} \left[\left(1 - \frac{1}{2i}\right) x_{i-1,j} + \left(1 + \frac{1}{2i}\right) x_{i+1,j} \right] + (1-M') x_{i,j} - \frac{b_2}{2} \Delta z r_{c,i,j} \right\} \quad (23)$$

$i = 1, 2, \dots, n-1$

迭代公式 (20) 和 (21) 可适用于对内部节点 ($i = 1, \dots, n-1$) 的迭代, 它们都有 $n-1$ 个方程, 但有 $n+1$ 个变量, 故还需分别补充两个方程, 才能求解。这可通过处理两个边界条件, 得到相应的两个代数方程, 从而使以上方程组可解。

③ 边界条件的处理

在管中心处 ($r=0$), 根据边界条件 (10) 和 (14) 及罗必塔 (L'Hopital) 法则, 可得

$$\lim_{r \rightarrow 0} \frac{\partial T / \partial r}{r} = \frac{\partial^2 T}{\partial r^2}, \quad \lim_{r \rightarrow 0} \frac{\partial x / \partial r}{r} = \frac{\partial^2 x}{\partial r^2}$$

分别代入方程 (4) 和 (5) 可得中心轴上的传热和传质方程

$$\frac{\partial T}{\partial z} - 2a_1 \cdot \frac{\partial^2 T}{\partial r^2} + b_1 r_c = 0 \quad (24)$$

$$\frac{\partial x}{\partial z} - 2a_2 \cdot \frac{\partial^2 x}{\partial r^2} - b_2 r_c = 0 \quad (25)$$

将式 (24) 和 (25) 写成 Crank-Nicholson 差分格式, 并考虑到反应管中心是轴对称的, 即

$$T_{-1,j} = T_{1,j}, \quad T_{-1,j+1} = T_{1,j+1}$$

$$x_{-1,j} = x_{1,j}, \quad x_{-1,j+1} = x_{1,j+1}$$

于是可得 $(1+2M)T_{0,j+1} = (1-2M)T_{0,j} + 2M(T_{1,j} + T_{1,j+1}) - \frac{b_1}{2} \Delta z (r_{c0,j} + r_{c0,j+1})$

$$(1+2M')x_{0,j+1} = (1-2M')x_{0,j} + 2M'(x_{1,j} + x_{1,j+1}) + \frac{b_2}{2} \Delta z (r_{c0,j} + r_{c0,j+1})$$

即 $T_{0,j+1} = F_{0,j} + \frac{1}{1+2M} \left(2MT_{1,j+1} - \frac{b_1}{2} \Delta z r_{c0,j+1} \right) \quad (26)$

$$x_{0,j+1} = G_{0,j} + \frac{1}{1+2M'} \left(2M'x_{1,j+1} + \frac{b_2}{2} \Delta z r_{c0,j+1} \right) \quad (27)$$

式中

$$F_{0,j} = \frac{1}{1+2M} \left[(1-2M)T_{0,j} + 2MT_{1,j} - \frac{b_1}{2} \Delta z r_{c0,j} \right] \quad (28)$$

$$G_{0,j} = \frac{1}{1+2M'} \left[(1-2M')x_{0,j} + 2M'x_{1,j} + \frac{b_2}{2} \Delta z r_{c0,j} \right] \quad (29)$$

式中 M 和 M' 同上。

右边界条件 (15) 的处理:

由于右边界条件是以导数形式给出的, 需要处理成为代数方程。通常需在 $r=a$ (即 $i=n$) 之外 Δr 处虚设一排点 $(n+1, j)$, 此排虚拟点与内点 $(n-1, j)$ 相对称。

由 $r=a$, $\partial x / \partial r = 0$ 得 $x_{n+1,j} = x_{n-1,j}$, $x_{n+1,j+1} = x_{n-1,j+1}$

代入 (21) 得 $x_{n,j+1} = G_{n,j} + \frac{1}{1+M'} \left(M'x_{n-1,j+1} + \frac{b_2}{2} \Delta z r_{cn,j+1} \right) \quad (30)$

$$\text{相应地, 由 (23) 得 } G_{n,j} = \frac{1}{1+M'} \left[M' x_{n-1,j} + (1-M') x_{n,j} + \frac{b_2}{2} \Delta z r_{cn,j+1} \right] \quad (31)$$

右边界条件 (11) 的处理:

用 Crank-Nicholson 方法导出在 $i=n$ ($r=a$) 处的差分格式, 即把

$$\begin{aligned} \frac{\partial T}{\partial z} \Big|_{n,j+\frac{1}{2}} &= \frac{T_{n,j+1} - T_{n,j}}{2(\Delta z/2)} = \frac{T_{n,j+1} - T_{n,j}}{\Delta z} \\ \frac{\partial T}{\partial r} \Big|_{n,j+\frac{1}{2}} &= \frac{1}{2} \left(\frac{\partial T}{\partial r} \Big|_{n,j} + \frac{\partial T}{\partial r} \Big|_{n,j-1} \right) = \frac{T_{n+1,j} - T_{n-1,j} + T_{n+1,j+1} - T_{n-1,j+1}}{4\Delta r} \end{aligned}$$

$$\text{代入式 (11): } \frac{T_{n,j+1} - T_{n,j}}{\Delta z} = C_1 \cdot \frac{T_{n+1,j} - T_{n-1,j} + T_{n+1,j+1} - T_{n-1,j+1}}{4\Delta r}$$

$$\text{即 } T_{n-1,j+1} = T_{n-1,j} + T_{n-1,j-1} - T_{n+1,j} + C_2 (T_{n,j+1} - T_{n,j}) \quad (32)$$

$$\text{式中, } C_2 = \frac{4\Delta r}{\Delta z C_1} \quad (33)$$

$$\text{把 (32) 代入 (16), 并整理得 } T_{n,j+1} = F_{n,j} + \frac{1}{C_3} \left(-M T_{n-1,j+1} + \frac{b_2}{2} \Delta z r_{cn,j+1} \right) \quad (34)$$

$$\text{式中 } F_{n,j} = \frac{1}{C_3} \left(-M T_{n-1,j} + C_4 T_{n,j} + \frac{b_1}{2} \Delta z r_{cn,j} \right) \quad (35)$$

$$C_3 = \frac{MC_2}{2} \left(1 + \frac{1}{2n} \right) - (1+M)$$

$$C_4 = \frac{MC_2}{2} \left(1 + \frac{1}{2n} \right) - (1-M)$$

这样, 由式 (20)、(21)、(26)、(27)、(30) 和 (34) 构成了迭代方程组, 初始值为

$$T_{i,0} = 873 \quad (i=0, 1, \dots, n-1), \quad T_{n,0} = 893$$

$$x_{i,0} = 873 \quad (i=0, 1, \dots, n)$$

增量取 $\Delta r = 0.01\text{m}$ (即 $n=5$), $\Delta z = 0.0585\text{m}$ 。

④ 平均转化率

所求解的是乙苯转化率为 45% 时所需的反应管长度。在某一长度 z 的平均转化率为

$$\bar{x} = \int_0^a r x dr / \int_0^a r dr$$

因为选取五个增量, 不能用辛普森法则积分, 但可用梯形法则:

$$\bar{x} = \frac{1}{25} (2x_{1,j} + 4x_{2,j} + 6x_{3,j} + 8x_{4,j} + 5x_{5,j}) \quad (36)$$

⑤ 程序说明 由 (20)、(21)、(26)、(27)、(30) 和 (34) 构成的线性方程组, 可用追赶法求解, 但这里调用 MATLAB 函数 `fsolve()` 进行求解。主程序中, 当平均转化率大于 0.45 时, 退出循环语句, 然后使用函数 `spline()` 插值计算平均转化率为 0.45 时所需的管长。

⑥ 程序清单 *PDEs2DS_CrankNicolson.m*

```
function PDEs2DS_CrankNicolson
global n F G rc dz M M1 b1 b2 C2 C3 C4
```

```

% r 和 z 方向的节点数及增量
n = 5+1; m = 100; % node number in r direction is 5
dr = 0.01; dz = 0.0585; % dr:m, dz: m

% 参数
Ramda = 0.45; G = 2500; Cp = 2.18; % Ramda: W/(m K), G: kg/(m^2 h), Cp: kJ/(kg K)
dH = 140e3; rho = 1440; a = 0.05; % dH: J/mol, rho: kg/m^3, a: radius of reactor, m
u0c0 = 0.069/(pi*a^2); % u0*c0 obtained from u0*c0*A=0.069 kmol/h
Cp1 = 1.0e3; F = 130/3600; % 烟道气比热: J/(kg K), F: kg/s

% 方程系数
a1 = Ramda/(G*Cp*1000)*3600; % m
b1 = dH*rho/G/Cp; % 注意单位统一
a2 = 0.000427; b2 = rho/u0c0; % a2 = D2/mu
C1 = 2*pi*a*Ramda/(F*Cp1); C2 = 4*dr/(dz*C1); % formula (12) 、 (33)
M = a1*dz/dr^2; M1 = a2*dz/dr^2; % formula (17)、 (19)
C3 = M*C2/2*(1+0.5/(n-1))-(1+M);
C4 = M*C2/2*(1+0.5/(n-1))-(1-M);

% (在反应器入口处即 z=0 或 j=1 时初值)
T(1:n-1) = 873; T(n) = 893; x(1:n) = 0; X0 = [T; x];
Tresult(1,:) = X0(1,:); xresult(1,:) = X0(2,:);
FGrc(T, x); % 计算对应于 j-1 时的 F(i, 1)和 G(i, 1)及反应速率 rc
for j=2:m
    X = fsolve(@TxEquations, X0); % 解非线性方程组
    T = X(1,:); x = X(2,:); Tresult(j,:) = T; xresult(j,:) = x
    xa(j,:) = sum(2*xresult(j, 2)+4*xresult(j, 3)+6*xresult(j, 4)...
        + 8*xresult(j, 5)+5*xresult(j, 6))/25
    if xa(j)>0.45, break, end % 当平均转化率>0.45 时, 退出循环
    FGrc(T, x) % 计算 F(i, j)和 G(i, j)及反应速度 rc(i, j)供下次迭代使用
end
z = dz.*[0:j-1]; r = dr.*[0:n-1];

% When the conversion rate of ethylbenzene is 45%, the reactor length (L) required is:
z = z'; L = spline(xa, z, 0.45)

% Plot the results
plot(z, xa), xlabel('z (m)'), ylabel('x_a_v')
figure, surf(r, z, Tresult), xlabel('r (m)'), ylabel('z (m)'), zlabel('T (K)')
figure, surf(r, z, xresult), xlabel('r (m)'), ylabel('z (m)'), zlabel('x')
% -----
function f = ReactionRate(T, x) % 计算反应速率
k = 0.027*exp(0.021*(T-773));
f = 15100*exp(-11000./T).*((1-x)./(1+x)-1.2*x.^2./k./(1+x).^2);
% -----

```

```

function f = FGrc(T, x) % Calculate F(i), G(i)
global F G rc n dz M M1 b1 b2 C2 C3 C4
rc = ReactionRate(T, x); % Calculate the reaction rate

F(1) = ((1-2*M)*T(1)+2*M*T(2)-b1*dz/2*rc(1))/(1+2*M); % 式(28)
G(1) = ((1-2*M1)*x(1)+2*M1*x(2)+b2*dz/2*rc(1))/(1+2*M1); % 式(29)
i = (2:5);
    var1 = 1-0.5/(i-1); var2 = 1+0.5/(i-1);
    F(i) = (M/2*(var1*T(i-1)+var2*T(i+1)) + (1-M)*T(i) - b1*dz/2*rc(i))/(M+1); % (22)
    G(i) = (M1/2*(var1*x(i-1)+var2*x(i+1)) + (1-M1)*x(i) + b2*dz/2*rc(i))/(M1+1); % (23)
F(n) = (-M*T(n-1) + C4*T(n) + b1*dz/2*rc(n))/C3; % 式(35)
G(n) = (M1*x(n-1) + (1-M1)*x(n) + b2*dz/2*rc(n))/(1+M1); % 式(31)
% -----
function f = TxEquations(X) % 由式(20)、(21)、(26)、(27)、(30)和(34)构成的线性方程组
global n F G rc dz M M1 b1 b2 C2 C3
T = X(1, :); x = X(2, :);
fT(1) = F(1) + (2*M*T(2)-b1/2*dz*rc(1))/(1+2*M) - T(1); % 式(26)
fx(1) = G(1) + (2*M1*x(2)+b2/2*dz*rc(1))/(1+2*M1) - x(1); % 式(27)
for i=2:n-1
    var1(i) = (1-0.5/(i-1)); var2(i) = (1+0.5/(i-1));
    fT(i) = F(i) + (M/2*(var1(i)*T(i-1)+var2(i)*T(i+1))-b1/2*dz*rc(i))/(M+1) - T(i); % (20)
    fx(i) = G(i) + (M1/2*(var1(i)*x(i-1)+var2(i)*x(i+1))+b2/2*dz*rc(i))/(M1+1) - x(i); % (21)
end
fT(n) = F(n) + (-M*T(n-1)+b1/2*dz*rc(n))/C3 - T(n); % 式(34)
fx(n) = G(n) + (M1*x(n-1)+b2/2*dz*rc(n))/(1+M1) - x(n); % 式(30)
f = [fT; fx];

```

⑦ 计算结果 平均转化率为 0.45 时所需的管长为 0.993 m。反应管轴径向温度分布如图 5-3 所示, 平均转化率沿管长分布如图 5-4 所示。运行该程序还可观察到轴径向平均转化率分布数据和分布图以及反应管轴径向温度分布数据等, 这里从略。

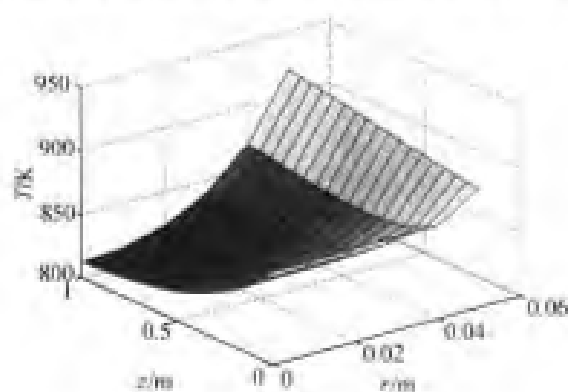


图 5-3 反应管轴径向温度分布

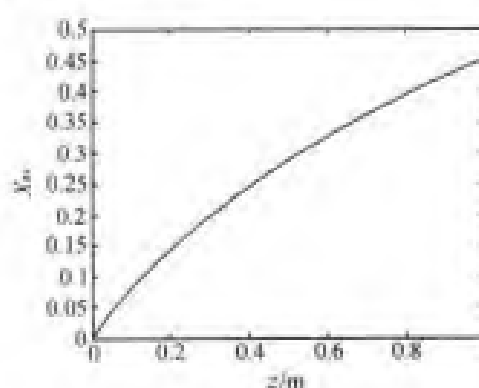


图 5-4 平均转化率沿管长的分布

5.2.4 二维动态 PDE 模型及其他问题

用有限差分求解二维动态 PDE 模型的方法与求解二维稳态 PDE 模型的方法一样，只是增加时间尺度上的离散而已，当然计算速度将明显变慢，具体步骤从略。

有关不规则边界和极坐标系统的有限差分，基本方法一样，只是针对不规则边界和极坐标系统做适当处理，详细可参考 Constantinides 和 Mostoufi (1999, p.427) [7]。

5.3 用正交配置法解偏微分方程

正交配置法

由于用正交配置法求解二维 PDE 问题的基本思想和具体方法与用正交配置法求解 ODE 边值问题完全类似，因此，为便于叙述，下面先介绍正交配置法用于求解常微分方程边值问题的具体方法和步骤，然后通过实例说明正交配置法在 PDE 求解方面的应用。

设常微分方程边值问题为 $F(y'', y', y, x) = 0$ (5-8a)

$y(0) = 0, y(1) = 1$ (5-8b)

对称问题的正交配置法：

取试函数为 $y = y(1) + (1-x^2) \sum_{i=1}^N a_i P_i(x^2)$ (5-9)

式中， a_i 为待定参数， N 为内配置点的个数， $P_N(x^2)$ 是 x^2 的正交多项式（见表 5-2）。

表 5-2 正交多项式 $P_N(x^2)$

	$N=1$	$N=2$	$N=3$
$a=1$ (平板)	$1-5x^2$	$1-14x^2+21x^4$	$1-27x^2+99x^4-\frac{429}{5}x^6$
$a=2$ (圆柱体)	$1-3x^2$	$1-8x^2+10x^4$	$1-15x^2+45x^4-35x^6$
$a=3$ (球体)	$1-\frac{7}{3}x^2$	$1-6x^2+\frac{33}{5}x^4$	$1-11x^2+\frac{143}{5}x^4-\frac{143}{7}x^6$

把式 (5-9) 写成通式，并把 $x=1$ 处的边界条件并入，得到

$$y = \sum_{i=1}^{N+1} d_i x^{2i-2} \quad i=1, 2, \dots, N+1 \quad (5-10)$$

由式 (5-10)，在多项式 $P_N(x^2)$ 的每一个根 x_j （配置点）处计算 y 、 y' 、 y'' 的值：

$$y(x_j) = \sum_{i=1}^{N+1} d_i x_j^{2i-2} \quad (5-11)$$

$$\left. \frac{dy}{dx} \right|_{x_j} = \sum_{i=1}^{N+1} 2(i-1) d_i x_j^{2i-3} \quad (5-12)$$

$$\left. \frac{d^2 y}{dx^2} \right|_{x_j} = \sum_{i=1}^{N+1} 2(i-1)(2i-3) d_i x_j^{2i-4} \quad (5-13)$$

上述方程可表示为如下的矩阵形式： $y = Qd$ (5-14)

$$\frac{dy}{dx} = Cd \quad (5-15)$$

$$\frac{d^2 y}{dx^2} = Dd \quad (5-16)$$

以上各式中, 向量 y 和 d 为: $y = [y(x_1), y(x_2), \dots, y(x_{N+1})]^T$

$$d = [d_1, d_2, \dots, d_{N+1}]^T$$

矩阵 Q 、 C 、 D 均为 $(N+1)$ 阶方阵, 其元素分别如下:

$$Q_{ji} = x_j^{2i-2} \quad (5-17)$$

$$C_{ji} = 2(i-1)x_j^{2i-3} \quad (5-18)$$

$$D_{ji} = 2(i-1)(2i-3)x_j^{2i-4} \quad (5-19)$$

式中, j 为行号 (等于根的号码), i 为列号。

由 (5-14) 可得 $d = Q^{-1}y$ (5-20)

代入 (5-15)、(5-16) 得 $\frac{dy}{dx} = Ay$ (5-21)

$$\frac{d^2y}{dx^2} = By \quad (5-22)$$

其中 $A = CQ^{-1}$ (5-23)

$$B = DQ^{-1} \quad (5-24)$$

若将式 (5-21) 和 (5-22) 中的第 j 行, 即对根 x_j 单独写出来, 可得:

$$\left. \frac{dy}{dx} \right|_{x_j} = \sum_{i=1}^{N+1} A_{ji} y(x_i) \quad (5-25)$$

$$\left. \frac{d^2y}{dx^2} \right|_{x_j} = \sum_{i=1}^{N+1} B_{ji} y(x_i) \quad (5-26)$$

对于长圆柱体 ($a=2$) 内配置点数为 3 时的 A_{ji} 和 B_{ji} 值列于表 5-3 中。对于平面体 ($a=1$) 和球体 ($a=3$) 内配置点数为 3 时的 A_{ji} 和 B_{ji} 值存于光盘中的文件“正交配置法的配置常数.doc”, 详细可参考 Finlayson^[2]、张建侯和许锡恩^[3]。

表 5-3 $P_N(x^2)$ 对于圆柱对称的配置常数—矩阵 A 和 B ($a=2$)

根的序号 j	A_{j1}	A_{j2}	A_{j3}	A_{j4}
1	3.359794	5.2924315	3.1010284	1.1683909
2	-1.3980385	-1.5627540	4.3197367	-1.3589422
3	0.69721650	-3.6766754	1.1267583	4.1062172
4	-1.2266754	5.4010626	-19.174383	15
根的序号 j	B_{j1}	B_{j2}	B_{j3}	B_{j4}
1	15.881426	19.636380	-5.2811862	1.5262327
2	11.151861	-34.497415	29.235709	-5.890155
3	-3.5405872	34.512110	-99.621159	68.649637
4	-33.869987	136.21969	-252.37970	150

对称配置法的计算步骤如下:

- ① 求出正交多项式 $P_N(x^2)$ 的 N 个零点 (内配置点) x_j ($j=1, 2, \dots, N$), 并令 $x_{N+1}=1$;
- ② 根据式 (5-17) ~ (5-19) 计算矩阵 Q 、 C 、 D , 再根据式 (5-23)、(5-24) 计算 A 和 B ;
- ③ 将各配置点 x_j ($j=1, 2, \dots, N$) 上的函数值、导数值即式 (5-25)、(5-26) 代入原方程 (5-8a), 得余量方程

$$F_j(y_1, y_2, \dots, y_N, x_j) = 0 \quad j = 1, 2, \dots, N;$$

- ④ 在 x_{N+1} 处, 由边界条件 (5-8b) 得: $y(x_{N+1}) = y(1)$;
 ⑤ 求解由 (5-10) 和 (5-11) 得到的代数方程组 (共 $N+1$ 个方程);
 ⑥ 由式 (5-20) 计算 d , 然后代入式 (5-10) 即得到近似解的表达式。

【例 5-4】 用对称正交配置法求解[例 5-3]的问题, 并与有限差分法的结果作比较。

解 由于原 PDE 问题是二维稳态方程组, 可以分别在两个空间方向 (z 方向和 r 方向) 使用正交配置方法离散, 将原 PDE 方程转变成代数方程组。另一种更简便的处理方法是, 只在 r 方向用正交配置方法离散, 使 PDE 方程变成 ODE 方程组, 然后调用龙格-库塔算法函数 ode45() 进行求解。

在 r 方向用对称正交配置法离散

要在 r 方向使用正交配置法, 需对它进行无因次化, 使无因次化后的对应自变量范围在区间 $[0, 1]$ 内。因此, 下面先将 r 无因次化, 即设 $R = \frac{r}{a}$, 则例 5-3 中的方程(4)、(5)和边界条件 (9~11)、(13~15) 变为:

$$\frac{\partial T}{\partial z} = \frac{a_1}{a^2} \left(\frac{\partial^2 T}{\partial R^2} + \frac{1}{R} \frac{\partial T}{\partial R} \right) - b_1 r_c \quad (\text{a})$$

$$\frac{\partial x}{\partial z} = \frac{a_2}{a^2} \left(\frac{\partial^2 x}{\partial R^2} + \frac{1}{R} \frac{\partial x}{\partial R} \right) + b_2 r_c \quad (\text{b})$$

$$z = 0, T = T_0, x = 0 \quad (\text{c})$$

$$R = 0, \frac{\partial T}{\partial R} = 0, \frac{\partial x}{\partial R} = 0 \quad (\text{d})$$

$$R = 1, \frac{\partial T}{\partial z} = \frac{C_1}{a} \frac{\partial T}{\partial R}, \frac{\partial x}{\partial R} = 0 \quad (\text{e})$$

取内配置点数为 N , 在 R 方向应用正交配置法, 即把

$$\left. \frac{\partial T}{\partial R} \right|_{x_j} = \sum_{i=1}^{N+1} A_{ji} T(R_i, z), \quad \left. \frac{\partial^2 T}{\partial R^2} \right|_{x_j} = \sum_{i=1}^{N+1} B_{ji} T(R_i, z)$$

代入 (a), 得到内配置点上的微分方程:

$$\frac{\partial T_j}{\partial z} = \frac{a_1}{a^2} \left(\sum_{i=1}^{N+1} B_{ji} T(R_i, z) + \frac{1}{R_j} \sum_{i=1}^{N+1} A_{ji} T(R_i, z) \right) - b_1 r_c \quad (\text{f})$$

$$\text{同样可得} \quad \frac{\partial x_j}{\partial z} = \frac{a_2}{a^2} \left(\sum_{i=1}^{N+1} B_{ji} x(R_i, z) + \frac{1}{R_j} \sum_{i=1}^{N+1} A_{ji} x(R_i, z) \right) + b_2 r_c \quad (\text{g})$$

$j = 1, 2, \dots, N$

式 (f) 和 (g) 共包含 $2 \times N$ 个方程, 而变量数为 $2 \times (N+1)$ 个, 因此还需根据 $R = 1$ (管径内壁) 处的边界条件 (e) 补充两个方程, 即在边界 $R = 1$ 处也用正交配置法, 式 (e) 变为:

$$\frac{\partial T_{N+1}}{\partial z} = \frac{C_1}{a} \sum_{i=1}^{N+1} A_{N+1,i} x(R_i, z) \quad (\text{h})$$

$$\frac{\partial x_{N+1}}{\partial z} = \sum_{i=1}^{N+1} A_{N+1,i} x(R_i, z) = 0 \quad (\text{i})$$

$$\text{由 (i) 得} \quad x_{N+1} = -\sum_{i=1}^N A_{N+1,i} x(R_i, z) / A_{N+1,N+1} \quad (\text{j})$$

$$\text{代入 (g), 得} \quad \frac{\partial x_j}{\partial z} = \frac{a_2}{a^2} \left(\sum_{i=1}^N (B_{ji} + A_{ji} / R_j) x(R_i, z) + (B_{N+1,j} + A_{N+1,j} / R_j) x(R_{N+1}, z) \right) + b_2 r_c \quad (\text{k})$$

至此, (f)、(h) 和 (k) 组成常微分方程组 (共 7 个方程)。

初值为 $T = (T_0, T_0, T_0, 893)$, $x = (0, 0, 0)$, 其中 $T_0 = 873$ 。

程序说明 只在 r 方向用正交配置法离散, 取三个内配置点。其中, `Euqations()` 定义由 (f)、(h) 和 (k) 组成的 ODE 方程组, `ReactionRate()` 用于计算反应速度, 主程序调用 `ode45()` 进行积分。

程序清单 *PDEs2DS_Collocation.m*

```
function PDEs2DS_Collocation % 只在 r 方向离散
clear all; clc
global A B a a1 a2 b1 b2 C1 C2 Rn T0 L N
% P3(x^2)对于圆柱对称 (a=2) 的配置常数 Aji 和 Bji 及全部配置点 Rn
A = [-3.3597940  5.2924315  -3.1010284  1.1683909
      -1.3980385  -1.5627540  4.3197367  -1.3589422
      0.69721650  -3.6766754  -1.1267583  4.1062172
      -1.2266754  5.4010626  -19.174383  15];
B = [-15.881426  19.636380  -5.2811862  1.5262327
      11.151861  -34.497415  29.2357090  -5.890155
      -3.5405872  34.512110  -99.621159  68.649637
      -33.869987  136.21969  -252.37970  150];
Rn = [0.29763730  0.63989598  0.88750181  1];
% 参数
Ramda = 0.45; G = 2500; Cp = 2.18; % Ramda: W/(m K), G: kg/(m^2 h), Cp: kJ/(kg K)
dH = 140e3; rho = 1440; a = 0.05; % dH: J/mol, rho: kg/m^3, a: radius of reactor
(m)
u0c0 = 0.069/(pi*a^2); % u0*c0 obtained from u0*c0*A=0.069 kmol/h
CP1 = 1.0*1e3; F = 130/3600; % CP1 即 CP: J/(kg K), F: kg/s
% 方程的系数
a1 = Ramda/(G*Cp*1000)*3600; % m
b1 = dH*rho/G/Cp; % Note of unit accordance: dH*rho/G/Cp rc = [K/m], rc=[kmol/(kg h)]
a2 = 0.000427; b2 = rho/u0c0; % a2 = D2/mu
C1 = 2*pi*a*Ramda/(F*CP1); % formula (12)
T0 = 873; L = 1.053; N = 3; % N: 内配置点个数
y0 = [T0 T0 T0 893 0 0 0]; % 向量 y = [T1 T2 T3 T4 x1 x2 x3]的初值
[z, y] = ode45(@Euqations, [0 L], y0); z
T = y(:, 1:N+1), x = y(:, N+2:2*N+1);
for i = 1:length(z), xb(i) = -sum(A(N+1, 1:N).* x(i, :)/A(N+1, N+1)); end
```

```

x = [x xb']
% 求沿管长的平均转化率 xa(i)
for i=1:length(z), xn = x(i, :); xa(i) = quadl(@func, 0, 1, [], [], Rn, xn)/(1^2/2), end
L = spline(xa, z, 0.45)

% Plot the results
surf(Rn*a, z, T), xlabel('r (m)'), ylabel('z (m)'), zlabel('T (K)') % 反应管轴径向温度分布
figure, plot(z, xa), xlabel('z (m)'), ylabel('x_a_v') % 平均转化率沿管长的分布图
figure, surf(Rn*a, z, x), xlabel('r (m)'), ylabel('z (m)'), zlabel('x') % 轴径向平均转化率分布
% -----
function dydx = Euqations(x, y)
global A B a a1 a2 b1 b2 C1 Rn T0 L N
T = y(1:N+1); x = y(N+2:2*N+1); rc = ReactionRate(T(1:N), x);
for i = 1:N, dTdR(i) = a1/a^2 * sum( (B(i, :)+A(i, :)/Rn(i)).* T') - b1*rc(i); end
dTdR(N+1) = C1/a*sum(A(N+1, :).* T')
xb = - sum(A(N+1, 1:N).* x'/A(N+1, N+1));
for i=1:N
    dxdR(i) = a2/a^2*( sum((B(i, 1:N)+A(i, 1:N)/Rn(i)).*x(1:N)) ...
        + (B(i, N+1)+A(i, N+1)/Rn(i)).*xb ) + b2*rc(i);
end
dydx = [dTdR dxdR]';
% -----
function f = ReactionRate(T, x) % 计算反应速率
k = 0.027*exp(0.021*(T-773));
f = 15100*exp(-11000./T).*((1-x)/(1+x)-1.2*x.^2./k./(1+x).^2);
% -----
function y = func(R, Rn, xn)
x = spline(Rn, xn, R);
y = R.*x;

```

计算结果 平均转化率为 0.45 时所需的管长为 0.994 m, 与[例 5-3]相同, 反应管轴径向温度分布、平均转化率沿管长的分布和轴径向转化率分布等结果, 都与[例 5-3]基本相同, 这里从略。

5.4 用 MOL 法解偏微分方程

MOL 法是将一个自变量当成连续变量, 而对其余的自变量用有限差分法或者正交配置法进行离散, 从而把偏微分方程转变为常微分方程组, 然后用龙格-库塔法积分求解。其中, MATLAB 提供的 MOL 法求解函数 `pdepe()` 可直接用来求解一维动态 PDE 方程(组), 它是采用正交配置法离散的。下面结合例子介绍用有限差分法离散的 MOL 法和 MATLAB 的 MOL 法求解函数 `pdepe()`。

5.4.1 MOL 法

先用有限差分法将 PDE 方程转变为差分-微分方程 (ODE 方程组), 再用龙格-库塔法积分。

【例 5-5】 用 MOL 法求解[例 5-1]的模型方程。

解 将原问题离散化

用有限差分代替空间导数, 得差分-微分方程

$$\frac{\partial T_m}{\partial t} = \frac{\alpha}{(\Delta x)^2} (T_{m-1} - 2T_m + T_{m+1}) \quad (\text{a})$$

取五个相等的增量 $\Delta x = 2 \text{ cm}$, 则由方程 (a) 得到关于 T_1 、 T_2 、 T_3 和 T_4 的四个常微分方程组:

$$\frac{\partial T_1}{\partial t} = \frac{\alpha}{(\Delta x)^2} (T_0 - 2T_1 + T_2) \quad (\text{b})$$

$$\frac{\partial T_2}{\partial t} = \frac{\alpha}{(\Delta x)^2} (T_1 - 2T_2 + T_3) \quad (\text{c})$$

$$\frac{\partial T_3}{\partial t} = \frac{\alpha}{(\Delta x)^2} (T_2 - 2T_3 + T_4) \quad (\text{d})$$

$$\frac{\partial T_4}{\partial t} = \frac{\alpha}{(\Delta x)^2} (T_3 - 2T_4 + T_5) \quad (\text{e})$$

其中, 由边界条件知, $T_0 = 100$, $T_5 = 0$ 。方程组 (b) ~ (e) 使用龙格-库塔法容易得到数值解。

程序说明 用 MOL 法求解一维动态偏微分方程: 先将 PDE 方程用有限差分法变为 ODE 方程, 然后用龙格-库塔法积分求解。程序取时间步长 (增量) 为 1s, 计算 $t = 8\text{s}$ 内的温度分布。

程序清单 PDE1Dd_MOL.m

```
Function PDE1Dd_MOL
global alpha dx Ta Tb
alpha = 2.0; dx = 2.0; Ta = 100; Tb = 0; tspan = [0 8]; T0 = [0 0 0 0];
[t, T] = ode45(@ODEs, tspan, T0)
% -----
function dTdt = ODEs(t, T)
global alpha dx Ta Tb
dTdt1 = Ta - 2*T(1) + T(2); dTdt2 = T(1) - 2*T(2) + T(3);
dTdt3 = T(2) - 2*T(3) + T(4); dTdt4 = T(3) - 2*T(4) + Tb;
dTdt = alpha/dx^2 * [dTdt1; dTdt2; dTdt3; dTdt4];
```

计算结果 $t = 8\text{s}$ 时对应于 $x = 0:2:10$ 的温度分布为 [100, 72.0444, 47.2326, 27.3612, 12.2526, 0]。

5.4.2 直接调用 MATLAB 函数 pdepe() 求解一维动态 PDE 方程 (组)

MATLAB 函数 pdepe() 可用于求解一维动态模型, 即一维抛物型和椭圆型 PDE 方程 (组) 的 IBVP 问题 (initial-boundary value problems, 初值-边值问题)。它用正交配置法进行空间离散。

下面先介绍 MATLAB 函数 pdepe() 及其配套使用的函数 pdeval(), 然后给出实例。

函数 pdepe()

函数 `pdepe()` 可求解的数学模型为:

$$c\left(x, t, u, \frac{\partial u}{\partial x}\right) \frac{\partial u}{\partial t} = x^{-m} \frac{\partial}{\partial x} \left(x^m f\left(x, t, u, \frac{\partial u}{\partial x}\right) \right) + s\left(x, t, u, \frac{\partial u}{\partial x}\right) \quad (5-27a)$$

$$t_0 \leq t \leq t_f, \quad a \leq x \leq b$$

式中, $m=0,1,2$ 分别代表平板、圆柱和球形。若 $m \geq 0$, 则要求 $a \geq 0$; $f(x, t, u, \partial u / \partial x)$ 为通量项 (flux term), $s(x, t, u, \partial u / \partial x)$ 为源项 (source term), $c(x, t, u, \partial u / \partial x)$ 为对角阵, 该对角阵的元素是零或者正数, 若元素是零, 则为椭圆型方程, 若是正数, 则为抛物型方程, 必须至少有一个方程是抛物型方程才可使用 `pdepe()` 求解。

初始条件 ($t=t_0$ 及所有 x): $u(x, t_0) = u_0(x)$ (5-27b)

边界条件 ($x=a$ 或 $x=b$ 及所有 t): $p(x, t, u) + q(x, t) f\left(x, t, u, \frac{\partial u}{\partial x}\right) = 0$ (5-27c)

注意 边界条件以通量项 $f(x, t, u, \partial u / \partial x)$ 表示, 而不用 $\partial u / \partial x$ 表示, 此外, 两个系数中, 只有 $p(x, t, u)$ 与 u 有关, 而 $q(x, t)$ 与 u 无关, $q(x, t)$ 的元素要么全为零, 要么都不为零。

函数 `pdepe()` 求解此类方程 (组) 的调用格式:

`sol = pdepe(m, pdefun, icfun, bcfun, xmesh, tspan)`

`sol = pdepe(m, pdefun, icfun, bcfun, xmesh, tspan, options)`

`sol = pdepe(m, pdefun, icfun, bcfun, xmesh, tspan, options, p1, p2...)`

输入变量:

m 表示问题的对称性, $m=0, 1, 2$ 分别代表平板、圆柱和球形;

pdefun 定义 PDE 问题的函数

icfun 定义初始条件的函数

bcfun 定义边界条件的函数

xmesh 指定网格点的向量 $[x_0, x_1, \dots, x_n]$, 其元素必须满足 $x_0 < x_1 < \dots < x_n$, **xmesh** 的长度必须为 3 以上。这样, 函数 `pdepe()` 将估算在此网格点上的 u_i 。

tspan 指定时间向量 $[t_0, t_1, \dots, t_f]$, 其元素必须满足 $t_0 < t_1 < \dots < t_f$, **tspan** 的长度必须为 3 以上。

options 默认积分参数可由 **options** 中的参数代替, ODE 求解器中只有一些参数可用于 `pdepe()` 中, 如: **RelTol**, **AbsTol**, **NormControl**, **InitialStep**, **MaxStep**。在大多数情况下, 用默认选项即可获得满意的解。参见 `odeset()`。

p1, p2... 可选参数, 可传递到 `pdefun()`、`icfun()` 和 `bcfun()` 中。

输出变量:

sol 函数返回方程的解 **sol** 是一个多维数组 $\text{sol}(j, k, i)$, 其中, 第 1 维代表时间 t_j , 第 2 维代表空间位置 x_k , 第 3 维代表解向量 **u** 的第 i 个元素 (因变量) u_i 。因此, 解向量 **u** 的第 i 个元素 (因变量) 的近似解为 $u_i = u_i = \text{sol}(:, :, i)$, 元素 $u_i(j, k) = \text{sol}(j, k, i)$ 是 u_i 在 $(t, x) = (t_{\text{span}}(j), x_{\text{mesh}}(k))$ 上的近似解。

函数 pdeval()

函数 `pdeval()` 常与函数 `pdepe()` 配套使用, 用于计算不在 **xmesh** 向量中的各点位置上的近似值及其偏导数值 $\partial u_i / \partial x$ 。

调用格式: `[uout, duoutdx] = pdeval(m, xmesh, ui, xout)`

输入变量: **m** 表示问题的对称性 (同前)。
 xmesh 网格点的向量 (同前)。
 ui $ui = \text{sol}(j, :, i)$ 为 PDE 在时间 $t(j)$ 和网格点 x 上的解, 其中 sol 是由函数 $\text{pdepe}()$ 计算返回的解。
 xout 在区间 $[x_0, x_n]$ 内的网格点向量, 函数 $\text{pdeval}()$ 将在此网格点上插值计算对应的解。
 输出变量: **uout** 对应于网格点 x_{out} 上的解。
 duoutdx 对应于网格点 x_{out} 上的偏导数值。

【例 5-6】 用 MATLAB 的 MOL 法函数 $\text{pdepe}()$ 求解[例 5-1]的模型方程:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \quad (\text{a})$$

I.C. 对 $t = 0$ 和 $0 \leq x \leq 10$, $T = 0^\circ\text{C}$ (b)

B.C. 对 $x = 0 \text{ cm}$ 和所有 t , $T = 100^\circ\text{C}$ (c)

 对 $x = 10 \text{ cm}$ 和所有 t , $T = 0^\circ\text{C}$ (d)

热扩散系数 $\alpha = 2.0 \text{ m}^2/\text{s}$ 。

程序说明

(i) 定义 PDE 问题的函数 $\text{pdefun}()$

将此 PDE 方程 (a) 与标准 PDE 方程 (5-27a) 比较, 可得

$$m = 0, \quad c = \frac{1}{\alpha}, \quad f = \frac{\partial T}{\partial x}, \quad s = 0$$

这样可容易写出 $\text{pdefun}()$ 。

(ii) 定义初始条件的函数 $\text{icfun}()$

为使程序更具通用性, 初始条件式 (b) 改写为

$$\text{对 } t = t_0 \text{ 和 } a \leq x \leq b, \quad T(x, t_0) = T_0 \quad (\text{e})$$

这里 $t_0 = 0$, $T_0 = 0^\circ\text{C}$, $a = 0$, $b = 10 \text{ cm}$

将此式与标准初始条件 (5-27b) 比较得, $T(x, t_0) = T_0(x) = T_0$, 这样可写出初始条件函数 $\text{icfun}()$ 。

(iii) 定义边界条件的函数 $\text{bcfunc}()$

考虑程序通用性, 边界条件式 (c) 和 (d) 改写为

$$\text{对 } x = a \text{ 和所有 } t, \quad T(a, t) = T_a \quad (\text{f})$$

$$\text{对 } x = b \text{ 和所有 } t, \quad T(b, t) = T_b \quad (\text{g})$$

式中, $T_a = 100^\circ\text{C}$, $T_b = 0^\circ\text{C}$

再变换成与边界条件 (5-27c) 相同的标准格式

$$(T(a, t) - T_a) + 0 \cdot f = 0$$

$$(T(b, t) - T_b) + 0 \cdot f = 0$$

然后与标准边界条件 (5-27c) 进行比较得,

$$pl = T(a, t) - T_a, \quad ql = 0$$

$$pr = T(b, t) - T_b, \quad qr = 0$$

这样即可写出边界条件函数 $\text{bcfunc}()$ 。

在定义 PDE 方程函数、初始条件函数和边界条件函数以后, 在程序中再给定网格向量和

时间向量，即可调用 `pdepe()` 进行求解。完整的程序为 `PDE1Dd_pdepe.m`，该程序还把数值解与解析解作比较。其中，该模型的解析解为： $T = 100.0 - 10.0x - \frac{200}{\pi} \sum_{n=1}^{\infty} \frac{1}{n} \sin \frac{n\pi x}{10} \exp\left(\frac{-n^2 \pi^2 \alpha t}{100}\right)$ 。

程序清单 `PDE1Dd_pdepe.m`

```
function PDE1Dd_pdepe % 用 MATLAB 函数 pdepe() 求解一维动态模型
global alpha ua ub
alpha = 2.0; ua = 100; ub = 0; m = 0; a = 0; b = 10; t0 = 0; tf = 8;
x = linspace(a, b, 6); t = linspace(t0, tf, 17);
sol = pdepe(m, @PDEfun, @ICfun, @BCfun, x, t);

% Extract the first solution component as u. This is not necessary
% for a single equation, but makes a point about the form of the output.
u = sol(:, :, 1), [m, nt] = size(t); nt = [1:1:nt], [m, nx] = size(x);

% analytical solution
n = [1:1000];
for i = 1:length(t)
    for j = 1:1:nx,
        u_analytical(i, j) = 100.0 - 10.0*x(j) - 200/pi*sum(sin(n.*pi* ...
            (j)/10).*exp(-n.^2*pi^2*alpha.*t(i)/100)./n)
    end
end
% surface plot of the solution
figure; surf(x, t, u); title('Numerical solution computed with 6 mesh points. ');
xlabel('Disuance x'); ylabel('Time t');

% solution profile at t=8
figure; plot(x, u(end, :), 'o', x, u_analytical(end, :)); title('Solutions at t = 8. ');
legend('Numerical, 6 mesh points', 'Analytical', 0); xlabel('Distance x'); ylabel('u(x, 8)');
% -----
function [c, f, s] = PDEfun(x, t, u, DuDx)
global alpha
c = 1/alpha; f = DuDx; s = 0;
% -----
function u0 = ICfun(x)
u0 = 0; % degree centigrade
% -----
function [pl, ql, pr, qr] = BCfun(xl, ul, xr, ur, t)
global ua ub
pl = ul-ua; ql = 0; pr = ur-ub; qr = 0;
```

计算结果 8s 内的温度分布如图 5-5 所示。

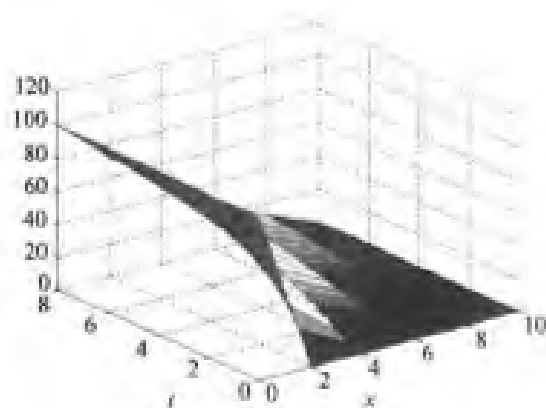


图 5-5 温度分布

将 Crank-Nicolson 方法、MOL 方法和 MATLAB 的 MOL 算法—`pdepe()` 的计算结果 (例 5-2、5-5、5-6) 列于表 5-4 中作比较。可见, 这三种方法的计算结果与解析解非常接近。

表 5-4 Crank-Nicolson 法、MOL 法和 MATLAB 的 MOL 算法—`pdepe()` 的计算结果与解析解的比较

例 5-6	解析解	100.0000	72.2312	47.4844	27.5520	12.3405	0.0000
例 5-2	Crank-Nicolson	100.0000	72.1210	47.3272	27.4209	12.2725	0.0000
例 5-5	MOL	100.0000	72.0444	47.2326	27.3612	12.2526	0.0000
例 5-6	PDEPE	100.0000	72.0464	47.2341	27.3604	12.2509	0.0000

注 显式差分法是不稳定的, 而 Crank-Nicolson 方法稳定有效。用 MOL 法可以减轻程序设计工作, 但为了避免数值不稳定性, 需要用小步长, 这可能耗费较多的计算时间。从函数 `pdepe()` 的使用说明来看, 该函数的算法实际上也是 MOL 法。

利用 `pdepe()` 还可以求解一维动态 PDE 方程组, 下面举例说明。

【例 5-7】 利用 MATLAB 函数 `pdepe()` 求解一维动态 PDE 方程组

$$\frac{\partial u_1}{\partial t} = c_1 \frac{\partial^2 u_1}{\partial x^2} - F(u_1 - u_2) \quad (a)$$

$$\frac{\partial u_2}{\partial t} = c_2 \frac{\partial^2 u_2}{\partial x^2} + F(u_1 - u_2) \quad (b)$$

$$0 \leq x \leq 1, \quad t \geq 0$$

$$\text{式中 } c_1 = 0.024, \quad c_2 = 0.170; \quad F(y) = \exp(5.73y) - \exp(-11.46y) \quad (c)$$

$$\text{I.C.} \quad u_1(x, 0) = 1, \quad u_2(x, 0) = 0 \quad (d)$$

$$\text{B.C.} \quad \frac{\partial u_1}{\partial x}(0, t) = 0, \quad u_2(0, t) = 0 \quad (e)$$

$$u_1(1, t) = 1, \quad \frac{\partial u_2}{\partial x}(1, t) = 0 \quad (f)$$

程序说明

(i) 定义 PDE 问题

将 PDE 方程组 (a)、(b) 改写成标准形式

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \frac{\partial}{\partial t} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \frac{\partial}{\partial x} \begin{bmatrix} c_1 \frac{\partial u_1}{\partial x} \\ c_2 \frac{\partial u_2}{\partial x} \end{bmatrix} + \begin{bmatrix} -F(u_1 - u_2) \\ F(u_1 - u_2) \end{bmatrix} \quad (g)$$

将 (g) 与标准方程 (5-27a) 比较得

$$m=0, \quad c=\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad f=\begin{bmatrix} c_1 \partial u_1 / \partial x \\ c_2 \partial u_2 / \partial x \end{bmatrix}, \quad s=\begin{bmatrix} -F(u_1 - u_2) \\ F(u_1 - u_2) \end{bmatrix}$$

由此定义 PDE 问题的函数 `pdex4pde()`。

(ii) 定义初始条件

由 I.C. 方程 (d) 得
$$u_0 = \begin{bmatrix} u_1(x, 0) \\ u_2(x, 0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

由此可定义初始条件的函数 `pdex4ic()`。

(iii) 定义边界条件

将 B.C. 方程 (e) 和 (f) 改写成标准形式

$$\begin{bmatrix} 0 \\ u_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} c_1 \partial u_1 / \partial x \\ c_2 \partial u_2 / \partial x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{for } x=0 \quad (\text{h})$$

$$\begin{bmatrix} u_1 - 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} c_1 \partial u_1 / \partial x \\ c_2 \partial u_2 / \partial x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{for } x=1 \quad (\text{i})$$

将 (h) 和 (i) 分别与标准式 (5-27b) 和 (5-27c) 比较, 得

$$pl = \begin{bmatrix} 0 \\ u_2(0, t) \end{bmatrix}, \quad ql = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$pr = \begin{bmatrix} u_1(1, t) - 1 \\ 0 \end{bmatrix}, \quad qr = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

由此可定义边界条件的函数 `pdex4bc()`。

在定义好 PDE、初始条件及边界条件后, 即可调用 `pdepe()` 求解。

程序清单 `pdex4.m` (此例取自 MATLAB 软件所附带的例子)

```
function pdex4
m = 0;
x = [0 0.005 0.01 0.05 0.1 0.2 0.5 0.7 0.9 0.95 0.99 0.995 1];
t = [0 0.005 0.01 0.05 0.1 0.5 1 1.5 2];

sol = pdepe(m, @pdex4pde, @pdex4ic, @pdex4bc, x, t);
u1 = sol(:, :, 1); u2 = sol(:, :, 2);

figure; surf(x, t, u1); title('u1(x, t)'); xlabel('Distance x'); ylabel('Time t');
figure; surf(x, t, u2); title('u2(x, t)'); xlabel('Distance x'); ylabel('Time t');
% -----
function [c, f, s] = pdex4pde(x, t, u, DuDx)
c = [1; 1]; f = [0.024; 0.17] .* DuDx;
y = u(1) - u(2); F = exp(5.73*y) - exp(-11.47*y); s = [-F; F];
% -----
function u0 = pdex4ic(x)
u0 = [1; 0];
% -----
function [pl, ql, pr, qr] = pdex4bc(xl, ul, xr, ur, t)
pl = [0; ul(2)]; ql = [1; 0]; pr = [ur(1)-1; 0]; qr = [0; 1];
```

【例 5-8】 用函数 pdepe() 求解多孔球形催化剂颗粒中的热质传递模型（一维动态 PDE 方程组）^[4]

在镍基催化剂作用下，苯催化加氢制环己烷的反应式为： $C_6H_6 + 3H_2 = C_6H_{12}$ ，所用的催化剂为多孔球形颗粒。试模拟计算催化剂颗粒中苯的浓度分布。

解 ① 数学模型

对球形催化剂颗粒进行质量平衡和热量平衡（传递方程）可得：

$$\varepsilon \frac{\partial C_B}{\partial t} = \frac{D_e}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial C_B}{\partial r} \right) - R_B \quad (\text{苯})$$

$$\varepsilon \frac{\partial C_H}{\partial t} = \frac{D_e}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial C_H}{\partial r} \right) - 3R_B \quad (\text{氢})$$

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\lambda_e}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial T}{\partial r} \right) + (-\Delta H) R_B$$

边界条件为：在 $r=0$ 处， $\frac{\partial C_B}{\partial r} = \frac{\partial C_H}{\partial r} = \frac{\partial T}{\partial r} = 0$

在 $r=1$ 处， $D_e \frac{\partial C_B}{\partial r} = k_g [C_B^0(t) - C_B]$

$$D_e \frac{\partial C_H}{\partial r} = k_g [C_H^0(t) - C_H]$$

$$\lambda_e \frac{\partial T}{\partial r} = h_g [T^0(t) - T]$$

初始条件为： $t=0$ 时， $C_B=0, C_H=0, T=T^0(0)$ ($0 \leq r \leq r_p$)

式中， C_B 和 C_H 分别为苯和氢气的浓度； T 为温度； D_e 为有效扩散系数（假定 B 和 H 相同）； λ_e 为有效导热系数。 ρ 为密度； c_p 为比热容； r 为径向坐标； r_p 为颗粒半径； t 为时间； k_g 为传质系数； h_g 传热系数； R_B 为苯的反应速率： $R_B = R_B(C_B, C_H, T)$ ；下标“0”代表周围环境的条件。

注意 C_B^0 ， C_H^0 和 T^0 是时间的函数，即它们随着反应器内的扰动而变化。

定义以下量纲为一参数： $y_B = \frac{C_B}{C_B^0(0)}$ ， $y_H = \frac{C_H}{C_H^0(0)}$ ， $\theta = \frac{T}{T^0(0)}$ ， $x = \frac{r}{r_p}$

则可得量纲为一模型：
$$\frac{\partial y_B}{\partial \tau} = \frac{1}{x^2} \frac{\partial}{\partial x} \left(x^2 \frac{\partial y_B}{\partial x} \right) - \Phi^2 \Re \quad (1)$$

$$\frac{\partial y_H}{\partial \tau} = \frac{1}{x^2} \frac{\partial}{\partial x} \left(x^2 \frac{\partial y_H}{\partial x} \right) - 3\Phi^2 \frac{C_B^0(0)}{C_H^0(0)} \Re \quad (2)$$

$$L_e \frac{\partial \theta}{\partial \tau} = \frac{1}{x^2} \frac{\partial}{\partial x} \left(x^2 \frac{\partial \theta}{\partial x} \right) + \beta \Phi^2 \Re \quad (3)$$

边界条件为：在 $x=0$ 处， $\frac{\partial y_B}{\partial x} = \frac{\partial y_H}{\partial x} = \frac{\partial \theta}{\partial x} = 0 \quad (4)$

在 $x=1$ 处， $\frac{\partial y_B}{\partial x} = Bi_m \left[\frac{C_B^0(t)}{C_B^0(0)} - y_B \right] \quad (5)$

$$\frac{\partial y_H}{\partial x} = Bi_m \left[\frac{C_H^0(t)}{C_H^0(0)} - y_H \right] \quad (6)$$

$$\frac{\partial \theta}{\partial x} = Bi_h \left[\frac{T^0(t)}{T^0(0)} - \theta \right] \quad (7)$$

$$\text{初始条件为: } \tau = 0 \text{ 时, } y_B = 0, y_H = 0, \theta = 1 \quad (0 \leq x \leq 1) \quad (8)$$

$$\text{式中, } \Phi^2 = \frac{r_p^2}{C_B^0(0)D_e} R_B(C_B^0(0), C_H^0(0), T^0(0)) \quad (\text{Thiele 模数的平方})$$

$$\beta = \frac{D_e(-\Delta H)C_B^0(0)}{\lambda_e T^0(0)} \quad (\text{Prater 数}), L_e = \frac{D_e \rho c_p}{\lambda_e \varepsilon} \quad (\text{Lewis 数})$$

$$\Re = \frac{R_B}{R_B(C_B^0(0), C_H^0(0), T^0(0))}, \quad \tau = \frac{D_e t}{r_p^2 \varepsilon} \quad (\text{无因次时间})$$

$$Bi_m = \frac{r_p k_g}{D_e} \quad (\text{传质 Biot 数}), \quad Bi_h = \frac{r_p h_g}{\lambda_e} \quad (\text{传热 Biot 数})$$

$$\text{对于苯加氢反应, 反应速率方程为: } R_B = \frac{\beta_{cat} k K \exp\left(\frac{Q-E}{R_g T}\right) P_B P_H}{1 + K \exp\left(\frac{Q}{R_g T}\right) P_B}$$

式中, $k = 3207 \text{ mol}/(\text{s} \cdot \text{atm}^0 \cdot \text{g 催化剂})$, $K = 3.207 \times 10^8 \text{ atm}^{-1}$, $Q = 16470 \text{ cal}^0/\text{mol}$, $E = 13770 \text{ cal/mol}$, $R_g = 1.9872 \text{ cal}/(\text{mol} \cdot \text{K})$, $\rho_{cat} = 1.88 \text{ g} \cdot \text{cm}^{-3}$, p_i 为组分 i 的分压。

$$\text{摩尔分数: } y_i = \frac{C_i}{C_i^0(0)} = \frac{P_i}{P_i^0(0)} \left(\frac{T^0(0)}{T} \right)$$

$$\Re = \exp \left[\alpha_2 \left(\frac{1}{\theta} - 1 \right) \right] \theta^2 y_B y_H \frac{[1 + K p_B^0(0) \exp(\alpha_1)]}{\left[1 + K p_B^0(0) \exp\left(\frac{\alpha_1}{\theta}\right) y_B \theta \right]}$$

$$\text{式中, } \alpha_1 = \frac{Q}{R_g T^0(0)}, \quad \alpha_2 = \frac{Q-E}{R_g T^0(0)}$$

② 程序说明

(i) 定义方程 记 $u = [y_B \quad y_H \quad \theta]^T$, 将式 (1) ~ (3) 与标准形 PDE 方程对照, 可得

$$C = [1 \quad 1 \quad Le]^T$$

$$m = 2$$

$$f = [DuDx(1) \quad DuDx(2) \quad DuDx(3)]^T$$

$$s = \begin{bmatrix} -\Phi^2 \Re & -3\Phi^2 \frac{C_B^0(0)}{C_H^0(0)} \Re & \beta \Phi^2 \Re \end{bmatrix}^T$$

由此定义方程函数 PDEs ()。

(ii) 初始条件 根据 (8) 式得: $u_0 = [0 \quad 0 \quad 1]^T$, 由此定义初始条件函数 IC()。

(iii) 边界条件 将此问题的边界条件 (4) ~ (7) 式与标准形边界条件对照, 可得:

$$\text{左边界 } (x=0): pl = [0 \quad 0 \quad 0]^T, \quad ql = [1 \quad 1 \quad 1]^T$$

$$\text{右边界 } (x=1): pr = [pr1 \quad pr2 \quad pr3]^T, \quad qr = [1 \quad 1 \quad 1]^T$$

① latm=101.325KPa。

② kcal=4.1840J。

$$\text{式中, } pr1 = -Bi_m \left[\frac{C_B^0(t)}{C_B^0(0)} - y_B \right], \quad pr2 = -Bi_m \left[\frac{C_H^0(t)}{C_H^0(0)} - y_H \right], \quad pr3 = -Bi_h \left[\frac{T^0(t)}{T^0(0)} - \theta \right]$$

由此定义边界条件函数 BC()。

这样, 程序调用 pdepe()即可求解此 PDE 方程组。

③ 程序清单 PDEsD1ds_Catalyst.m

```
function PDEsD1ds_Catalyst % 用 pdepe()求解三个一维动态 PDE 方程组
global K PB0 Le fai alph1 alph2 CB0 CH0 beta Bim Bih
k = 3207; K = 3.207e-8; Q = 16470; % k: mol/(s atm gcat), K: atm-1, Q: cal/mol
E = 13770; Rg = 1.9872; rho_cat = 1.88; % E: cal/(mol K), Rg: cal/(mol K), rho_cat: g/cm3
% Parameter data for catalyst start-up simulation
fai = 1.0; beta = 0.04; % fai: Thiele modulus, beta: Prater number
Le = 80; Bim = 350; % Le: Lweis number, Bim: mass Biot number
Bih = 20; CB0 = 0.025; % Bih: heat Biot number, CB0: initial concentration of benzene
CH0 = 0.975; T0 = 373.15; % CH0: initial concentration of hydrogen, T0: initial temperature
alph1 = Q/(Rg*T0); alph2 = (Q-E)/(Rg*T0); PB0 = CB0*Rg*T0; % 苯的初始压力
m = 2; h = 0.125/2; x = [0:h:1]; t = [0:0.01:1];

sol = pdepe(m, @PDEs, @IC, @BC, x, t);
u1 = sol(:, :, 1); % u1 represents yB (Dimensionless concentration of benzene)
u2 = sol(:, :, 2); % u2 represents yH (Dimensionless concentration of Hydrogen)
u3 = sol(:, :, 3); % u3 represents theta (Dimensionless temperature)

% Calculate yB at t=0.1 and at the position xout = [0.00 0.25 0.50 0.75 1.00]
xout = [0.00 0.25 0.50 0.75 1.00];
for i=1:5
    yBout(i) = u1(find(t==0.1), find(x==xout(i)));
end
disp('yB at t=0.1 and at the position xout = [0.00 0.25 0.50 0.75 1.00]:', disp(yBout))

% Plot yB~x at t=0.01, 0.1 and 1
figure; plot(x, u1(find(t==0.01), :), 'r--', x, u1(find(t==0.1), :), 'b', x, u1(find(t==1), :), 'k-');
xlabel('x'); ylabel('y_B'); legend('t=0.01', 't=0.1', 't=1');

% Plot y_B(x, t),
figure; surf(x, t, u1); title('y_B(x, t)'); xlabel('x'); ylabel('t');
% -----
function [c, f, s] = PDEs(x, t, u, DuDx) % For solution of the PDE system with pdepe()
global K PB0 Le fai alph1 alph2 CB0 CH0 heta
numer = 1 + K*PB0*exp(alph1); denom = 1 + K*PB0*exp(alph1/u(3))*u(1)*u(3);
R = exp(alph2*(1/u(3)-1))*u(3)^2*u(1)*u(2)*numer/denom;
c = [1; 1; Le]; f = [DuDx(1); DuDx(2); DuDx(3)]; s = fai^2*R*[-1; -3*CB0/CH0; beta];
% -----
```

```

function u0 = IC(x);                                % 定义初始条件
u0 = [0; 0; 1];
% -----
function [pl, ql, pr, qr] = BC(xl, ul, xr, ur, t)    % 定义边界条件
global Bim Bih
pl = [0; 0; 0];   ql = [1; 1; 1];
pr = [-Bim*(1-ur(1)); -Bim*(1-ur(2)); -Bih*(1-ur(3))];   qr = [1; 1; 1];

```

④ 计算结果 $\tau = 0.1$ 时苯的浓度分布 y_B 数据如下。

x	0.00	0.25	0.50	0.75	1.00
$h = 0.125$	0.2677	0.3271	0.4917	0.7361	0.9973
$h = 0.125/2$	0.2680	0.3265	0.4922	0.7371	0.9972
$h = 0.125/4$	0.2682	0.3260	0.4923	0.7375	0.9972

$\tau = 0.01, 0.1, 1$ 时催化剂颗粒中苯的浓度分布曲线 y_B-x 如图 5-6, 苯浓度随空间位置和时间变化的变化 $y_B(x, t)$ 如图 5-7。

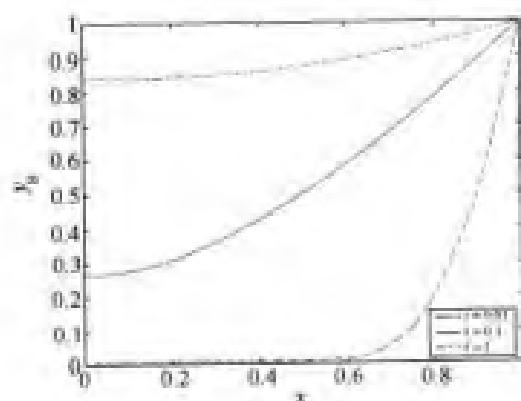


图 5-6 $\tau = 0.01, 0.1, 1$ 时苯的浓度分布曲线 y_B

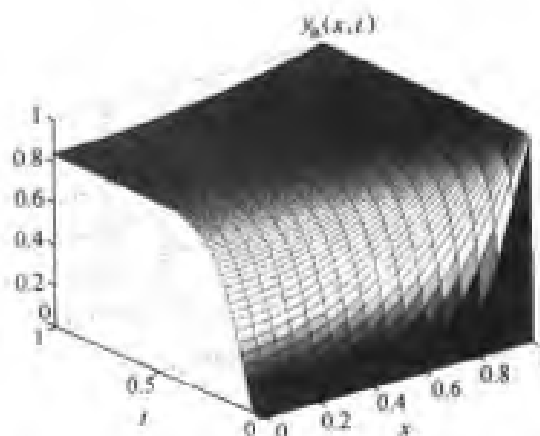


图 5-7 催化剂颗粒中苯的浓度随位置和时间变化的变化

5.5 用有限元法解偏微分方程

5.5.1 MATLAB PDE 工具箱简介

有限元法是近十几年发展起来的一种很有效的计算方法。MATLAB PDE 工具箱正是一个用有限元法求解二维线性和非线性偏微分方程的强大工具。它能够求解以下四大类 PDE 方程:

椭圆型 (Elliptic) 偏微分方程 (通用 Poisson 方程)

$$-\nabla \cdot c \nabla u + au = f \quad \text{in } \Omega \quad (5-28a)$$

$$-\nabla \cdot (c \otimes \nabla u) + au = f \quad \text{in } \Omega \quad (5-28b)$$

式中 c , a 和 f 可以是常数, 也可以是位置 (x, y) 、因变量 u 及其导数 $\partial u / \partial x$ 和 $\partial u / \partial y$ 的函数, 符号 ∇ 为向量微分符号 (下同), Ω 是平面上的有界区域 (下同), 式 (5-28a) 为单个 PDE 方程, 而式 (5-28b) 包含 “ \otimes ” 表示该式为 PDE 方程组 (下同)。

对于非线性椭圆型 PDE 方程

$$-\nabla \cdot (c(u)\nabla u) + a(u)u = f(u) \quad \text{in } \Omega \quad (5-28c)$$

$$-\nabla \cdot (c(u) \otimes \nabla u) + a(u)u = f(u) \quad \text{in } \Omega \quad (5-28d)$$

式中, c 、 a 和 f 是未知 u 的函数。PDE 工具箱提供专门的非线性 PDE 求解器, 即 `pdenonlin()`。

抛物型 (Parabolic) 偏微分方程 (组)

$$d \frac{\partial u}{\partial t} - \nabla \cdot c \nabla u + au = f \quad \text{in } \Omega \quad (5-29a)$$

$$d \frac{\partial \mathbf{u}}{\partial t} - \nabla \cdot (c \otimes \nabla \mathbf{u}) + a\mathbf{u} = \mathbf{f} \quad \text{in } \Omega \quad (5-29b)$$

双曲型 (Hyperbolic) 偏微分方程 (组) (波动方程)

$$d \frac{\partial^2 u}{\partial t^2} - \nabla \cdot c \nabla u + au = f \quad \text{in } \Omega \quad (5-30a)$$

$$d \frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla \cdot (c \otimes \nabla \mathbf{u}) + a\mathbf{u} = \mathbf{f} \quad \text{in } \Omega \quad (5-30b)$$

方程 (5-29a) ~ (5-30b) 中的系数 c 、 a 和 f 可以是关于 (x, y) 及时间 t 的复杂函数。

本征型 (特征值) PDE 问题 (系统)

$$-\nabla \cdot c \nabla u + au = \lambda u \quad (5-31a)$$

$$-\nabla \cdot (c \otimes \nabla \mathbf{u}) + a\mathbf{u} = \lambda \mathbf{u} \quad (5-31b)$$

以上各方程都是二维的, 所以 $\nabla \cdot c \nabla u$ 表示为

$$\nabla \cdot c \nabla u = \frac{\partial}{\partial x} c \frac{\partial u}{\partial x} + \frac{\partial}{\partial y} c \frac{\partial u}{\partial y} \quad (5-32)$$

边界条件 在 PDE 工具箱中, 边界条件有三种形式:

(1) Dirichlet 型边界条件 $hu = r$ (5-33)

式中系数 h 是一个标量 (通常 $h = 1$), r 为因变量 u 在边界 Ω 上的数值。

(2) 广义 Neumann 型边界条件 $\mathbf{n} \cdot c \nabla u + qu = c \frac{\partial u}{\partial n} + qu = g$ (5-34)

式中, \mathbf{n} 是外法线向量, 而 c 是 PDE 方程中的系数, 系数 q 和 g 为常数或特定的函数。

(3) 混合边界条件 即 Dirichlet 型边界条件和广义 Neumann 型边界条件的组合:

$$\begin{cases} hu = r \\ \mathbf{\bar{n}} \cdot c \nabla u + qu = c \frac{\partial u}{\partial n} + qu = g + h'l \end{cases} \quad (5-35)$$

注意 PDE 工具箱中广义 Neumann 边界条件和混合边界条件这两个术语与 5.1.2 中所介绍的不同, PDE 工具箱中的广义 Neumann 边界条件相当于 5.1.2 节中混合边界条件, 而 PDE 工具箱中的混合边界条件, 在 5.1.2 中是没有涉及的。因此, 在用 PDE 工具箱求解 PDE 问题时, 应采用 PDE 工具箱所用的边界条件术语。

以上方程 (5-28a) ~ (5-31b) 中, 各个 PDE 方程组 (包含符号“ \otimes ”的方程) 都具有上述的三种边界条件, 而单个 PDE 方程的边界条件只有 Dirichlet 型边界条件和广义 Neumann 型边界条件两种。换句话说, 混合边界条件 (5-35) 只适用于 PDE 方程组。

初始条件 抛物型 PDE 方程: $u(t_0) = u_0(x)$ (5-36)

双曲型 PDE 方程: $u(t_0) = u_0(x)$ (5-37a)

$u'(t_0) = u'_0(x)$ (5-37b)

用 PDE 工具箱求解 PDE 问题的两种方法

有两种方式可以使用 PDE 工具箱来求解偏微分方程, 一种是使用 `pdetool`, 另一种是使用命令行函数通过编程进行 PDE 求解。前者是傻瓜式求解方式, 无需编程, 只需执行 `pdetool` 命令, 通过 `pdetool` 图形用户界面, 就可以交互式地进行 PDE 问题的几何模型建立、边界条件设定、PDE 方程类型设置、三角形网格划分及网格细化、参数给定、方程求解和图形显示, 等等 (见 5.5.3)。后者通过编程调用 PDE 工具箱中的函数来求解 PDE 问题。当 PDE 问题比较复杂或不具备 `pdetool` 规定的标准形式时, 可以使用命令行函数编程求解 (见 5.5.4 节)。

柱坐标中 PDE 问题的求解

尽管 PDE 工具箱基本上是针对 x - y 坐标中的问题求解的, 但轴对称 r - z 柱坐标 PDE 问题, 可通过变换而“伪装”成 x - y 坐标中的标准 PDE 问题, 如已知 r - z 柱坐标 PDE 问题为

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + \frac{\partial^2 u}{\partial z^2} = f(u) \quad (5-38a)$$

由式 (5-38a) 变为

$$\frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) + \frac{\partial}{\partial z} \left(r \frac{\partial u}{\partial z} \right) = rf(u) \quad (5-38b)$$

令 $x \equiv r$, $y \equiv z$, 则 (5-38b) 式变为

$$\frac{\partial}{\partial x} \left(x \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(x \frac{\partial u}{\partial y} \right) = xf(u) \quad (5-38c)$$

即

$$-\nabla \cdot x \nabla u = -xf(u) \quad (5-38d)$$

对照 (5-28a) 式可知, (5-38d) 式是标准的 PDE 形式, 故可用 MATLAB PDE 工具箱求解。

5.5.2 PDE 工具箱可解决的化学工程问题

5.5.2.1 传热 (Heat Transfer)

典型例子: 二维稳态热传导 (椭圆型 PDE 方程)、二维非稳态热传导 (抛物形 PDE 方程)。

通用传热 PDE 方程:

(1) 平面轴对称的非稳态传热过程为抛物线型 PDE 方程

$$\rho C \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q + h(T_{\text{ext}} - T) \quad (5-39)$$

式中, ρ 为密度; C 为热容; k 为导热系数; Q 为热源; h 为对流传热系数; T_{ext} 为外部温度。方程最右边项 $h(T_{\text{ext}} - T)$ 是与周围环境交换的热量。

(2) 对于平面轴对称的稳态传热系统, 可得到椭圆型传热方程

$$-\nabla \cdot (k \nabla T) = Q + h(T_{\text{ext}} - T) \quad (5-40)$$

5.5.2.2 扩散传递 (Diffusion Transfer) —— 传质

典型例子: 反应器的二维稳态模型和二维动态模型 (扩散+反应)。

通用扩散 (传质) PDE 方程具有与传热方程相同的结构:

(1) 平面轴对称的非稳态扩散过程也是抛物线型 PDE 方程

$$\frac{\partial c}{\partial t} - \nabla \cdot (D \nabla c) = Q \quad (5-41)$$

式中, c 为浓度; D 为扩散系数; Q 为体积源。

(2) 对于平面轴对称的稳态扩散系统, 可得到椭圆型扩散方程

$$-\nabla \cdot (D \nabla c) = Q \quad (5-42)$$

5.5.2.3 流体力学方程

(1) 势流 (potential flows) 和多孔介质中的流动 (porous medium flows) 以 Laplace 方程表示:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0, \quad \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0, \quad \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 0 \quad (5-43)$$

(2) 考虑轴向速度分布的管内层流流动 (Laminar flow), 涉及 Poisson 方程:

$$\frac{\partial^2 v_z}{\partial x^2} + \frac{\partial^2 v_z}{\partial y^2} = \frac{1}{\mu} \frac{dp}{dz} \quad (5-44)$$

其中, x, y 是任意横截面的平面坐标, z 是流体流动方向

方程 (5-28a) 也可表示为

$$-\nabla \cdot c \nabla u + au = -\left(\frac{\partial}{\partial x} c \frac{\partial u}{\partial x} + \frac{\partial}{\partial y} c \frac{\partial u}{\partial y} \right) + au = f \quad (5-45)$$

可见, 方程 (5-43) 和 (5-44) 实际上是通用 Poisson 方程 (5-28a) 的特例。

边界条件 主要有两种形式, 即前面所述的 Dirichlet 型和 Neumann 型:

(1) Dirichlet 型 $hu = r$ (5-33)

式中系数 h 是一个标量因子 (通常 $h = 1$), r 可设为一个常数或一个函数。一个典型的 Dirichlet 型边界条件是: $h = 1$ 、 r 为因变量在边界上的数值。

(2) Neumann 型 $\mathbf{n} \cdot c \nabla u + qu = c \frac{\partial u}{\partial n} + qu = g$ (5-34)

式中, \mathbf{n} 是外法线向量, 而 c 是 PDE 方程中出现的系数。用户可以选择系数 q 和 p 为常数 (通常为 0), 或特定的函数。如果 g 和 q 为 0, 则 Neumann 条件变为

$$\frac{\partial u}{\partial n} = 0 \quad (5-46)$$

对于 $u = \phi$ (势流函数, potential function) 或 $u = p$ (压力, 在多孔介质中的应用), (5-46) 式表示沿法线方向无流体通过边界; 对于 $u = \psi$ (stream function), (5-46) 式表示流体完全地沿法线方向通过边界。

PDE 工具箱在流体力学中的代表性应用

可用 PDE 工具箱求解的流体力学问题汇总于如表 5-5, 其各种边界条件如表 5-6 所示。

表 5-5 PDE 工具箱在流体力学问题中的代表性应用

应 用	u 等同于	典 型 参 数
势流	ϕ	$c = 1, a = f = 0$
势流	ψ	$c = 1, a = f = 0$
旋流 (Rotational flow)	ψ	$c = 1, a = 0, f = \zeta$ (vorticity)
多孔介质中的流动	p	$c = \kappa$ (permeability), $a = f = 0$
管内层流流动	v_z	$c = 1, a = 0, f = (-dp/dz)/\mu$

表 5-6 流体力学问题的各种边界条件

应 用	无流体通过边界条件	有流体流过边界条件
势流, ϕ	$\partial \phi / \partial n = 0$	$u = r$
势流, ψ	$\psi = r$	$\partial \psi / \partial n = 0$

续表

应 用	无流体通过边界条件	有流体流过边界条件
旋流, ψ	$\psi = r$	$\partial\psi/\partial n = 0$
多孔介质中的流动, p	$\partial p/\partial n = 0$	$p = r$
管内层流流动, u_z	在管壁, $u_z = 0$	

5.5.3 利用 pdetool 求解偏微分方程

MATLAB PDE 工具箱可以解决具有二维空间变量和一个时间变量的偏微分方程, 它所提供的 pdetool 能够帮助用户灵活而高效地求解此类问题。利用此图形用户界面, 可以交互式地进行偏微分方程数学问题的几何模型建立、边界条件设定、偏微分方程类型设置、三角形网格划分及网格细化、参数给定、方程求解和图形显示等。

5.5.3.1 利用 pdetool 求解 PDE 问题的一般步骤

在 MATLAB 命令窗口中执行 pdetool 命令, 即显示 PDE 工具箱的图形用户界面(图 5-8)。

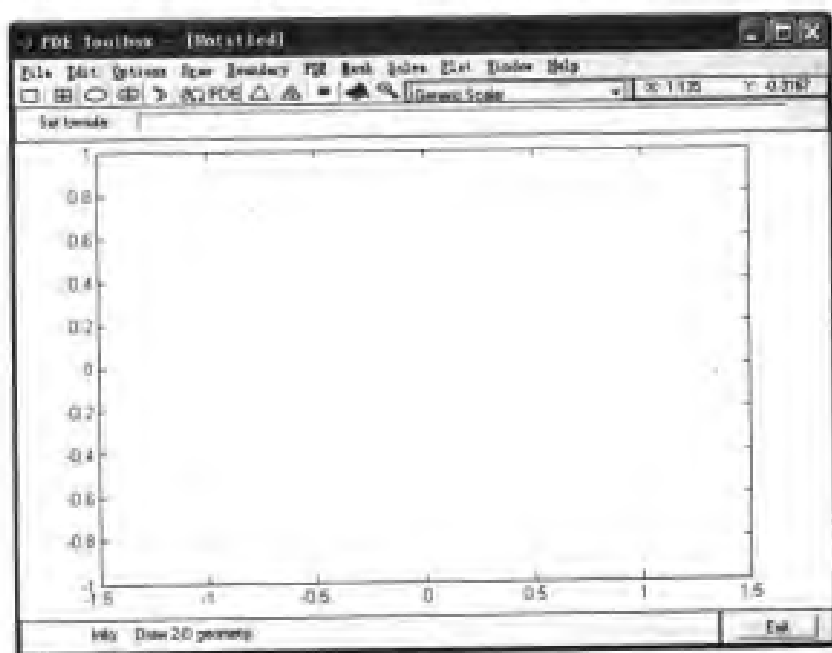


图 5-8 PDE 工具箱的图形用户界面

菜单除了 File 和 Edit 外, 从左到右依次为 Options、Draw、Boundary、PDE、Mesh、Solve 和 Plot, 其顺序正好与用 PDE 工具箱求解偏微分方程的步骤一致, 即按如下步骤求解。

- ① 用 Options (选项) 菜单设置应用模式;
- ② 用 Draw 菜单建立几何模型;
- ③ 用 Boundary 菜单设定边界条件;
- ④ 用 PDE 菜单定义偏微分方程的类型和方程的系数;
- ⑤ 用 Mesh (网格) 菜单进行三角形网格划分及网格细化;
- ⑥ 用 Solve 菜单进行偏微分方程的求解;
- ⑦ 用 Plot 以图形方式显示结果。

下面结合[例 5-9]详细说明如何使用 pdetool 求解 PDE 方程的具体步骤。

【例 5-9】 用 pdetool 求解二维稳态——椭圆型 PDE 边值问题 (Poisson 方程)

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2, & (-5 < x < 5, -5 < y < 5) \\ u = 100, & (x = \pm 5, y = \pm 5) \end{cases}$$

详细的求解步骤如下。

(1) 用 Options 菜单设置应用模式

PDE 工具箱提供多种应用模式，用户应根据实际的 PDE 问题选择合适的应用模式。

用鼠标选中 Options 菜单的 Application 子菜单，或者直接在工具条中单击 Generic Scalar 下拉式列表框，即弹出多种应用模式供选择，如图 5-9 所示。

可供选择的应用模式有：

Generic Scalar	一般标量模式
Generic System	一般系统模式
Structural Mechanics, Plane Stress	结构力学平面应力应用模式
Structural Mechanics, Plane Strain	结构力学平面张力应用模式
Electrostatics	静电学应用模式
Magnetostatics	磁静电学应用模式
AC Power Electromagnetics	交流电电磁学应用模式
Heat Transfer	传热应用模式
Diffusion	扩散应用模式

其中，Generic Scalar、Generic System、Heat Transfer 和 Diffusion 这四种应用模式，对化学工程有用。具体地说，对于传热方面的单个 PDE 方程，选择 Heat Transfer 应用模式，对于传质扩散方面的单个 PDE 问题，选择 Diffusion 应用模式。当然这两种情况，都可选用 Generic Scalar 应用模式来求解，只是因变量名不是 T 或 C，且方程系数 a、c、f、d 没有任何物理意义，使用不那么方便。对于具有两个方程的二维偏微分方程组，只能选择 Generic System 应用模式。

本例使用 Generic Scalar（默认）模式，并选择 Options 菜单的 Axes Limits 子菜单以设置坐标范围： $x = [-6.6]$ ， $y = [-6.6]$ ，如图 5-10 所示。

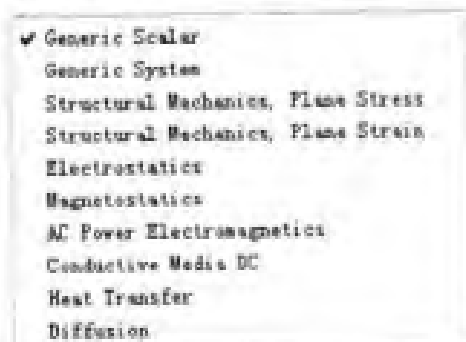


图 5-9 Application 弹出式菜单图








5-10 设置坐标范围

在 Options 菜单中选择 Grid 选项，可在图中显示网格。单击 Grid Spacing...选项，打开 Grid Spacing 对话框，可以调整网格间隔的大小。选择 Snap 选项，则鼠标在图中点击时，将自动选择离该点最近的网格交点。在画图之前设置上面的选项，将使绘图更加方便。

(2) 利用 Draw 绘制几何模型

用 Draw 菜单中的画图功能或工具条中的前五个画图工具按钮,即可画出 PDE 问题的几何模型(求解域的几何形状)。这五个画图工具按钮代表的意义如下:

- | | | |
|---|--|---|
|  绘矩形或方形 |  绘同心矩形或方形 |  绘椭圆或圆 |
|  绘同心椭圆或圆 |  绘多义线 | |

在本例中,先画一个矩形图,再单击该矩形图,弹出如图 5-11 的设置对话框,可在此对话框中设置矩形的位置及大小。设置好确认后得到如图 5-12 所示的几何模型。

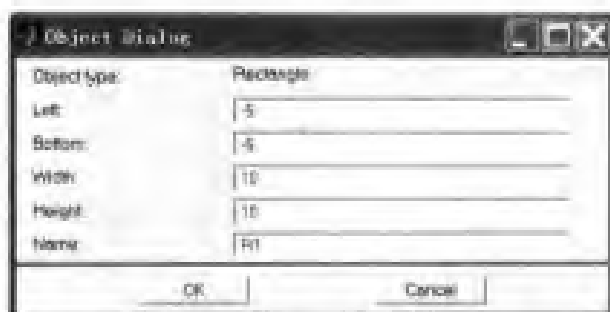


图 5-11 设置对话框

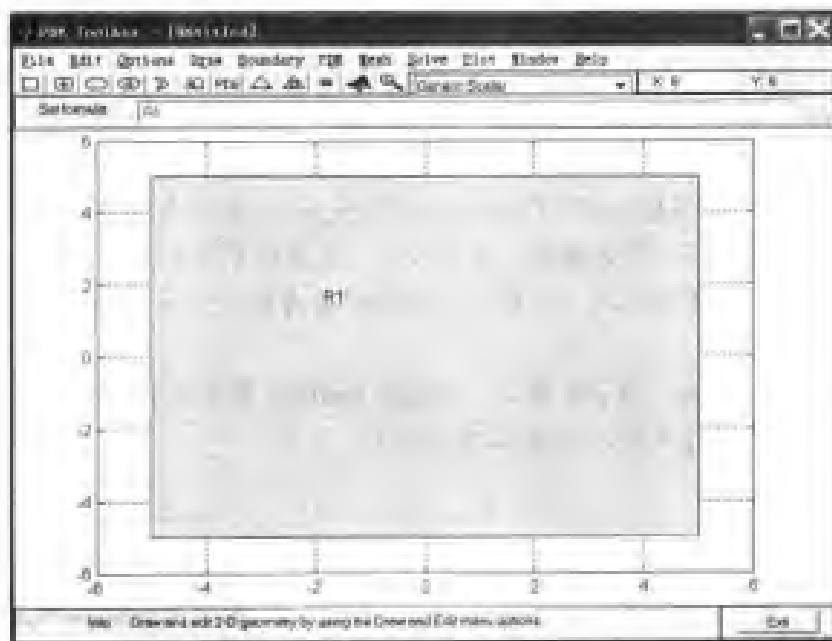



图 5-12 几何模型

(3) 定义边界条件

先在 Boundary 菜单中选中 Boundary mode 选项或直接单击工具栏上的  按钮,即显示几何模型的边界,如图 5-13 所示。再选择要定义的边界,即单击图中边界,使边界线变成黑色(表示它被选中)。或按住 Shift 键,以连续选择多条边界线段。然后,双击边界或在 Specify Boundary Conditions...选项,即弹出 Boundary Condition(边界条件)设定对话框。对不同的应用模型,弹出的边界条件对话框也不同,这里由于前面选择了 Generic Scalar 模型,因此弹出如图 5-14 所示的对话框。

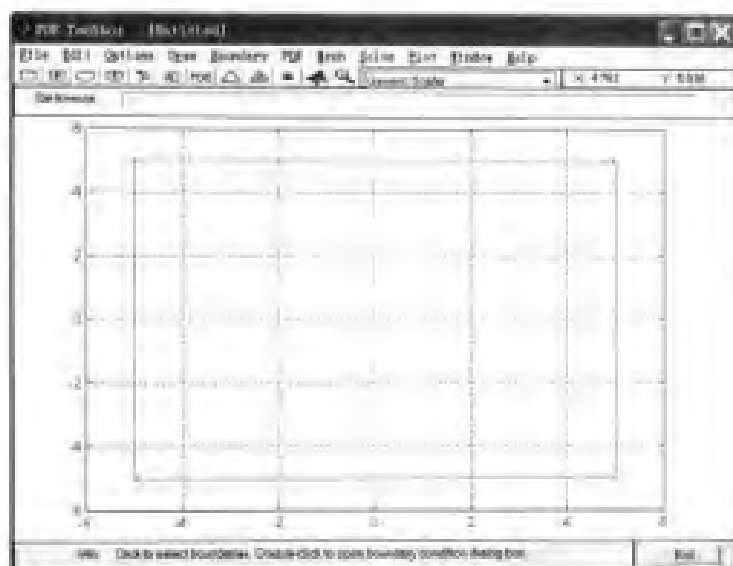


图 5-13 几何模型的边界

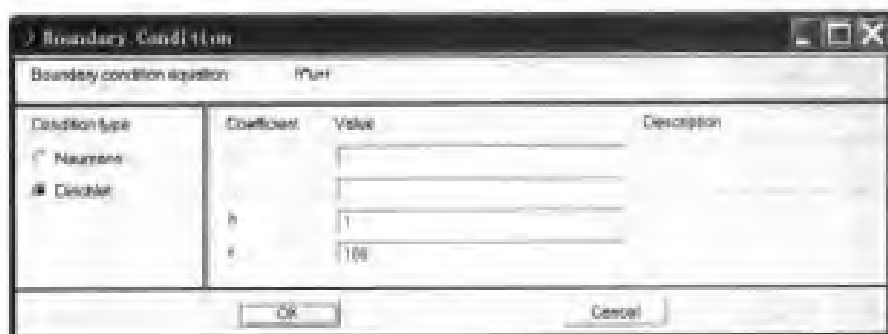


图 5-14 边界条件对话框

在对话框中选择边界条件的类型，即选择 Neumann 或 Dirichlet 边界条件。

该对话框包括三个区域：

Boundary condition equation 标签：显示边界条件方程。

Condition type（边界条件类型）：

Neumann 单选按钮 — 选择该项时表示将边界条件设置为 Neumann 型边界条件；

Dirichlet 单选按钮（默认） — 选择该项时表示将边界条件设置为 Dirichlet 型边界条件。

Coefficient Value 栏 — 该栏文本框用于输入边界条件公式中的系数值。

本例中选择默认设置（Dirichlet 边界条件），并设置 $h=1$ ， $r=100$ ，即 $u=100$ 。

定义好边界条件后，还可以把几何模型和边界条件分别保存为几何模型的 M 文件和边界条件的 M 文件，以供命令行函数求解 PDE 方法使用。具体步骤是：先执行 Boundary->Export Decomposed Geometry, Boundary cond'，弹出如图 5-15 的对话框。

按[OK]键即把几何模型和边界条件数据（g 和 b）输出到 MATLAB 工作环境，然后切换到工作环境并执行下面命令：

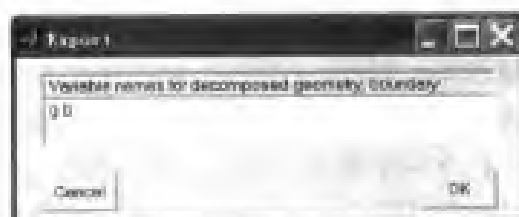


图 5-15 几何条件和边界条件数据输出对话框

```
>>wgeom(g, 'poissong')
```

```
>>wbound(b, 'poissonb')
```

则把几何模型和边界条件的有关数据 (g 和 b) 保存为几何模型的 M 文件 poissong.m 和边界条件的 M 文件 poissonb.m。这两个 M 文件可供程序调用以求解 PDE。

可见, pdetool 还可作为命令行函数求解 PDE 方法的辅助工具, 根据以上三个步骤 1.1~1.3 生成几何模型 M 文件和边界条件 M 文件, 减少了命令行函数求解 PDE 方法的许多编程工作量。

(4) 定义 PDE 类型并输入方程系数

在 PDE 菜单中选中 PDE mode 选项, 图形即显示为 PDE 模式。若再选中 Show Subdomain Labels 选项, 则将在图中显示子域标签。在 PDE 菜单中选中 PDE specification 选项或直接单击工具栏上的 PDE 按钮, 即弹出 PDE specification 的对话框, 如图 5-16 所示。

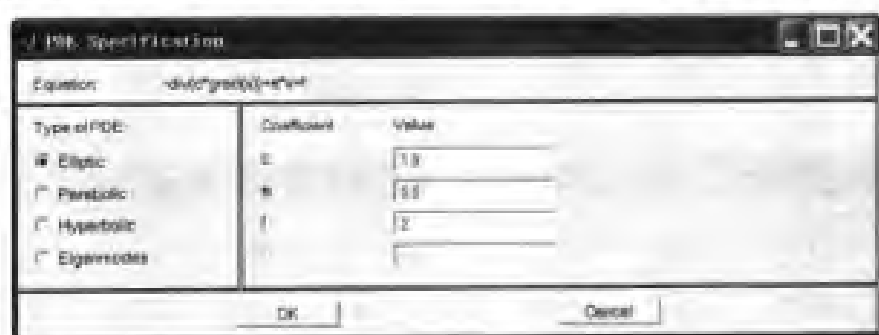


图 5-16 PDE 方程设置对话框

在此对话框中可以设置偏微分方程的类型及方程系数。对话框中各选项意义如下:

Equation 标签 显示 PDE 方程

Types of PDE 栏 该栏包括以下四个单选按钮, 选择其中一个以确定 PDE 问题。

Elliptic 设置为椭圆型 PDE 方程 (默认选项);

Parabolic 设置为抛物型 PDE 方程;


Hyperbolic 设置为双曲型 PDE 方程;



Eigenmodes 设置为特征值 PDE 方程;

Coefficient Value 栏 在该栏文本框中设置 PDE 方程的系数值。

本例中选择 Elliptic 方程 (默认设置), 方程系数为: $c=1.0$, $a=0.0$, $f=2$ 。

(5) 三角形网格划分及网格细化

在 Mesh 菜单中选中 Initialize mesh 选项, 或在工具栏中单击  按钮, 即可对求解域进行三角形网格初始化。如图 5-17 所示。

在 Mesh 菜单中选中 remesh 选项, 或在工具栏中单击  按钮, 即可对初始网格进行精细化。多次执行 remesh 功能或多次单击  按钮, 即进行多次的网格精细化工作, 这将得到更精细的网格, 从而提高求解精度。但网格越细, 计算时间越长, 所以网格的精细化次数要适当。本例执行 remesh 后的精细化网格如图 5-18 所示。

在 Mesh 菜单中选中 Jiggle mesh 选项, 即可对网格进行微调。选择 Display Triangle Quality, 将显示三角形的质量图。如图 5-19 所示。图中, 三角形的最好质量定为 1, 并以红色表示, 最差质量为 0, 以蓝色表示, 二者之间的三角形质量根据质量系数的大小 (0~1) 用

红色和蓝色之间的过渡色表示。

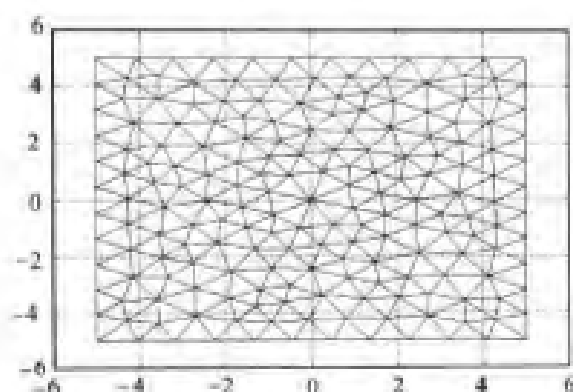
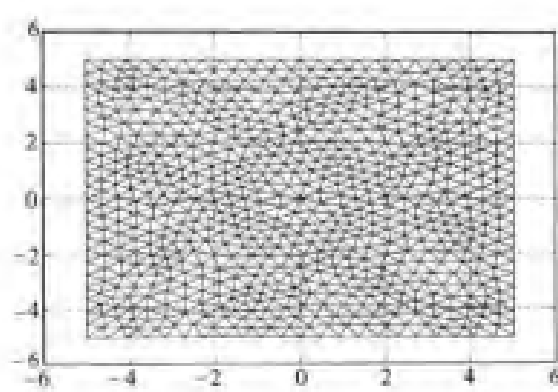


图 5-17 初始化网格图



5-18 精细化的网格

在 Mesh 菜单中选择 Show node labels 选项，将在图中显示网格节点的编号。选择 Show subdomain labels 选项，将显示各子域的编号。

选择 Parameters... 选项，将打开 Mesh Parameters 对话框，如图 5-20 所示。在此对话框中可以确定与网格划分有关的参数。对话框中各控件的意义如下：

Maximum edge size 文本框	输入数值以确定最大边缘大小
Mesh growth rate 文本框	输入网格生长率，默认值为 3:1
Jiggle mesh 核选框	选中此项，则网格可微调。默认时选中此项。
Jiggle mode 下拉列表	确定微调模式，有 On, Optimize minimum 和 Optimize mean 等三项供选择。
Number of jiggle iterations 文本框	输入微调迭代的次数
Refinement method 下拉列表	确定网格精细化的方法，有 regular 和 longest 两个选项可供选择。

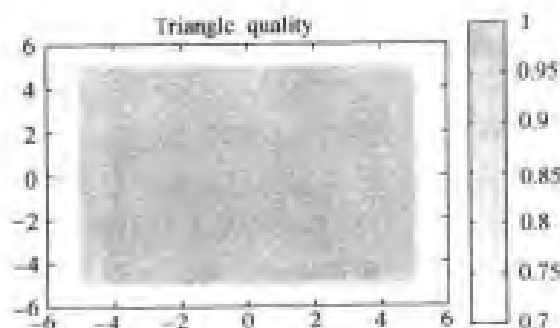


图 5-19 微调网格

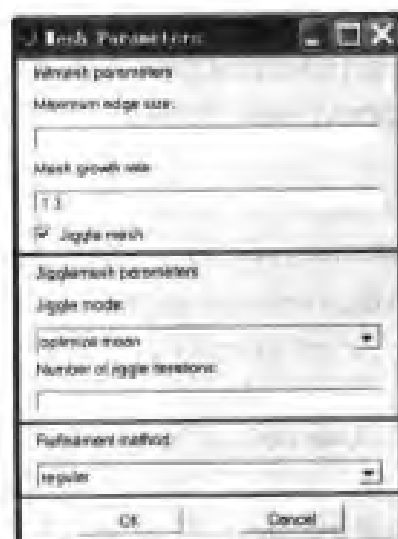


图 5-20 Mesh Parameters 对话框图

执行 [Mesh]->[Export Mesh...], 即弹出如图 5-21 所示的对话框，按 [OK] 键即把网格数据

输出到 MATLAB 工作环境中。

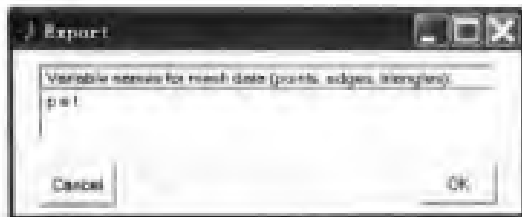



图 5-21 输出网格数据的对话框

(6) PDE 方程的求解

在 Solve 菜单中选择 Solve PDE 选项，或在工具栏中单击  按钮，即可对前面定义的 PDE 问题进行求解。这时得到默认解，如图 5-22 所示。

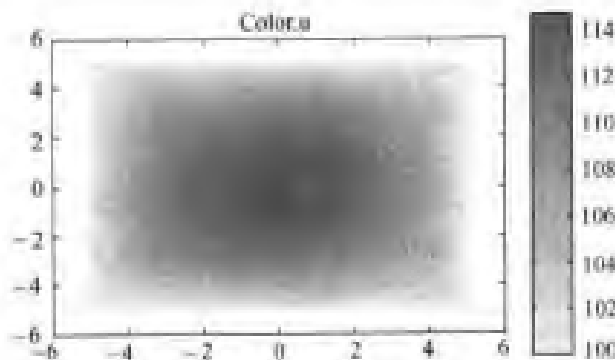


图 5-22 默认图解（色谱图）

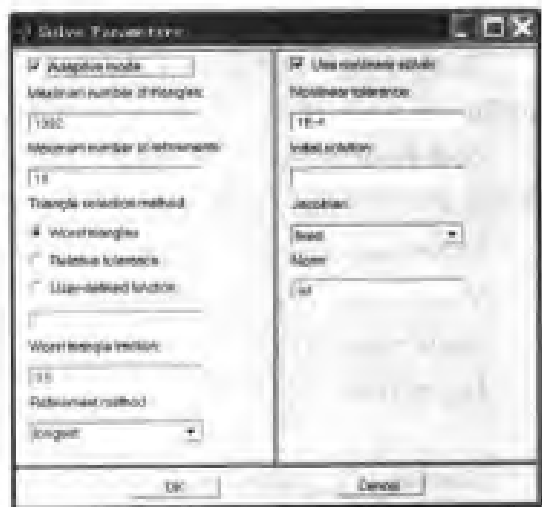


图 5-23 Solve Parameters 对话框

在 Solve 菜单中单击 Parameters... 选项，打开 Solve Parameters 对话框，如图 5-23 所示。在此对话中可以设置求解方法和参数。该对话框中各选项的意义如下：

Adaptive mode 核对框

选中此项，即选择自适应模式，系统将自适应生成网格并进行求解。

Maximum number of triangles 文本框

用于输入三角形的最大个数

Maximum number of refinements 文本框

用于输入网格精细化的最大次数

Triangle selection method

Worst triangles

选择此项，即根据最坏三角形进行选择

Relate tolerance

选择此项，即根据相对误差进行选择

User-defined function

选择此项，并在下面的文本框中输入用户自定义的函数，即可根据该函数选择三角形

Worst triangle fractions 文本框

用于输入最坏三角形分数

Refinement method 下拉列表

用于选择网格精细化的方法，有 regular 和 longest 两个选项供选择。

User nonlinear solver 核选框

选中此项，即用非线性求解器求解

Nonlinear tolerance 文本框

用于输入非线性迭代终止的误差限。默认值为 $1e^{-4}$ 。

Initial solution 文本框

用于输入 PDE 问题的初值


Jacobian 下拉列表

用于选择确定雅可比矩阵的方式，共有 fixed、lumped 和 full 可供选择。

Norm 文本框

用于输入范数，其默认值为 $\text{Inf}(\infty)$

(7) 解的图形显示

pdetool 可直接将 PDE 的数值解以丰富多彩的图形方式显示出来，默认为彩色图。在 Plot 菜单中单击 Parameters... 选项，或单击  按钮，即弹出 Plot Selection 对话框，如图 5-24 所示。

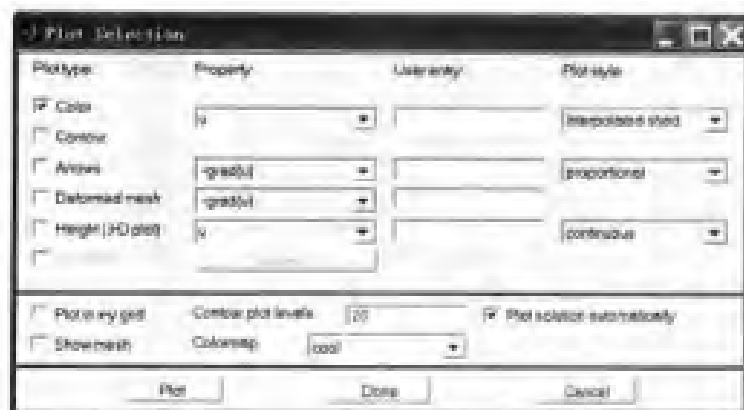


图 5-24 Plot Selection 对话框

该对话框中各选项的意义如下：

Plot type 控件列 该列控件提供控制图形类型的选择，包括：

Color 选中此项，则可生成并显示解的彩色图。默认时选中此项。

Contour 选中此项，则可生成并显示解的等值线图。

Arrows 选中此项，则可生成并显示解的矢量图。

Deformed mesh 选中此项，则可生成并显示解的变形网格图。

Height (3-D plot) 选中此项，则可生成并显示解的三维图。

Animation 选中此项，则可生成并显示解的动画。

Property 控件列 其所有控件都是下拉列表，用于定义对解的哪一部分进行图形显示。

User entry 控件列 该列所有控件都是文本框，用于用户输入。

Plot style 控件列 该列所有控件都是下拉列表，用于控制前面图形的不同风格。

Plot in x-y grid 核选框 选中此项，将在 x-y 网格中绘图。

Show mesh 核选框 选中此项，将显示网格。

Contour plot levels 文本框 用于输入等直线的水平线。

Colormap 下拉列表 用于选择绘制彩色图的颜色。

Plot solution automatically 核选框 选中此项，系统将自动绘制解的图形。

图 5-25、图 5-26 和图 5-27 分别是选取 Plot type 中的 Contour、Arrows 和 Height (3-D plot) 时得到的图形。

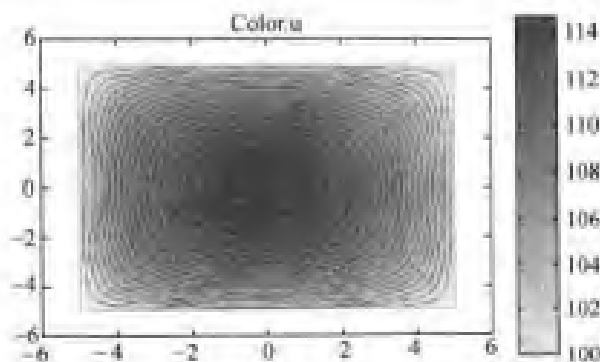


图 5-25 色谱图叠加等值线

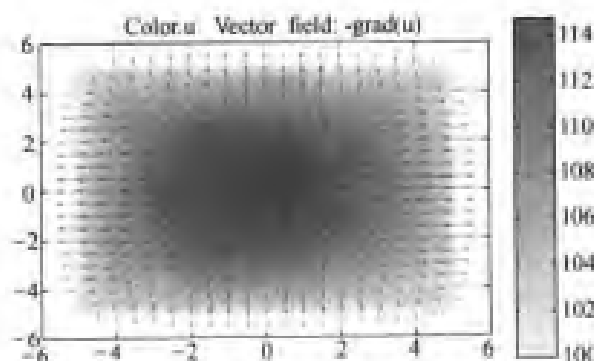


图 5-26 色谱图叠加矢量

(8) 输出数值解

在 Solve 菜单中选择 Export Solution，弹出如图 5-27 的对话框，按[OK]键即把数值解 u 输出到 MATLAB 工作环境中，再切换到工作环境（命令方式）中，对 u 进行显示和处理，见图 5-28。

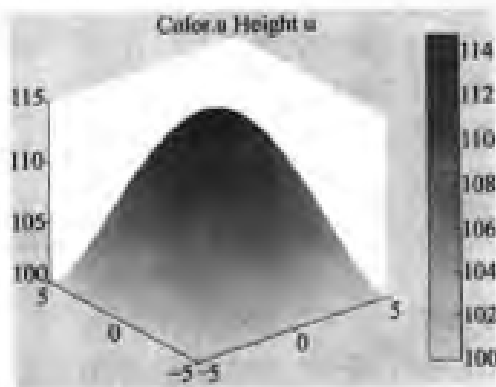


图 5-27 三维图

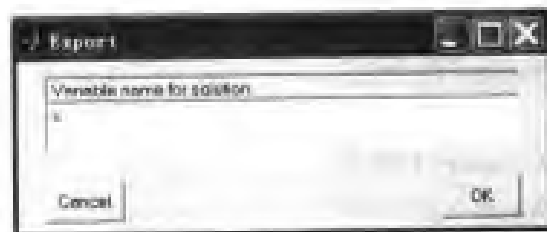


图 5-28 输出数值解的对话框

最后，存入文件 *PDEtoolPoisson.m* 中。

5.5.3.2 应用实例

【例 5-10】 用 *pde tool* 求解二维非稳态热传导方程（无内热源）——抛物型 PDE 方程

有一块受热的有矩形裂缝的金属块，金属块的左侧被加热到 100°C ，在右侧热量从金属块以恒定速率传给周围空气，其他所有的边界都绝热。相应的方程及边界条件为

$$d \frac{\partial u}{\partial t} - \Delta u = 0 \quad (u \text{ 为温度, } d \text{ 为系数})$$

B.C.

$$u = 100 \quad (\text{左侧, Dirichlet 条件})$$

$$\frac{\partial u}{\partial n} = -10 \quad (\text{右侧, Neumann 条件})$$

$$\frac{\partial u}{\partial n} = 0 \quad (\text{其他边界, Neumann 条件})$$

I.C.

$$t = 0, u = 0$$

试计算 5 s 时的温度分布。

解 执行 pdetool 命令打开其界面, 并按下列步骤求解。

(1) 用 Options 菜单定义应用模式

选择 Options 菜单中 Application 子菜单中的 Generic Scalar (通用标量) 应用模式。为方便下一步几何模型画图, 选中 Options 菜单中的 Grid 功能, 再单击 Grid Space 打开其对话框 (图 5-29), 在 X-axis extra ticks 文本框中输入“-0.05 0.05”, 这样将在 x 轴-0.05 和 0.05 处绘制附加的线, 以帮助画出代表裂缝的狭窄矩形。然后选中 Snap 功能。

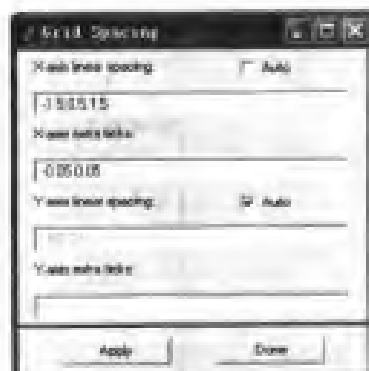



图 5-29 Grid Spacing 对话框

(2) 用 Draw 画 CSG 模型

先画一个矩形 (R1), 其四角的坐标为 $x = [-0.5, 0.5, 0.5, -0.5]$, $y = [-0.8, -0.8, 0.8, 0.8]$ 。再画另一个矩形 (R2) 代表矩形裂缝, 其四角坐标为 $x = [-0.05, 0.05, 0.05, -0.05]$, $y = [-0.4, -0.4, 0.4, 0.4]$ 。画好两个矩形后, 为清晰起见可先把 [Grid] 功能关闭, 再在图形上方的 Set formula 文本框中把“R1+R2”改为“R1-R2”, 然后单击  按钮, 即完成几何模型的绘制 (图 5-30)。

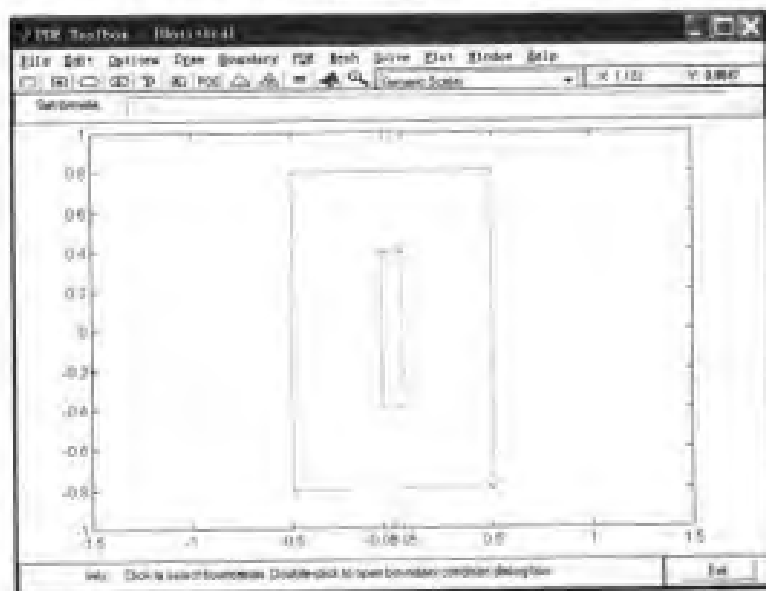


图 5-30 几何模型

(3) 定义边界条件

双击左边界, 选中“Dirichlet”并设置 $h = 1$, $t = 100$, 如图 5-31 所示, 再按 [OK] 键确认。

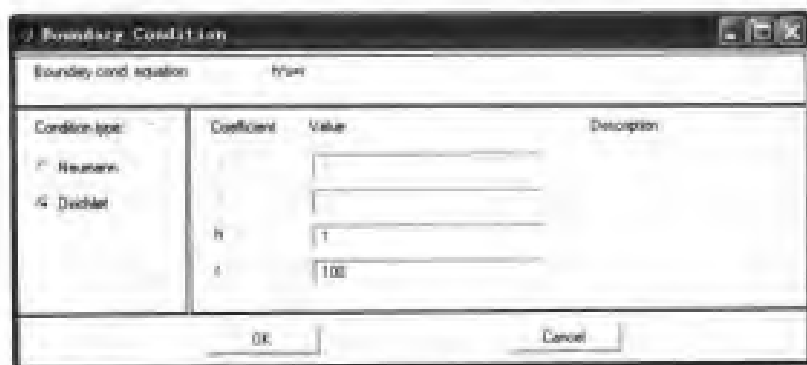


图 5-31 边界条件设置

双击右边界，选中“Neumann”并设置 $g = -10$ ， $q = 0$ ，如图 5-32 所示，再按[OK]键确认。

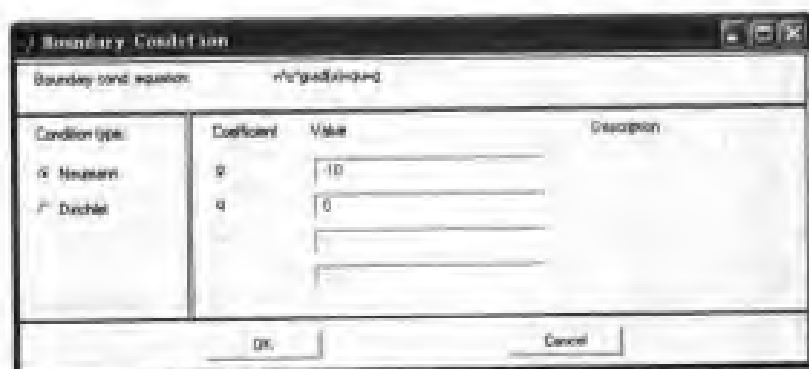


图 5-32 边界条件设置

剩下的六条边界，由于它们的边界条件方程相同，可先按住[Shift]键，同时用鼠标依次单击这六条边界，这样它们同时被选中，这时松开[Shift]键，再双击其中一条边界，在弹出的边界条件对话框中选中“Neumann”，并设置 $g = 0$ ， $q = 0$ ，再按[OK]键确认，即完成全部六条边界的设置。

至此，所有边界条件已经全部设置完毕。

(4) 定义 PDE 方程及方程系数

先选中 PDE 菜单中的 PDE mode 功能，再单击工具栏中的 PDE 按钮，即弹出 PDE specification 的对话框，选 Parabolic，并设置 $c = 1$ ， $a = 0$ ， $f = 0$ ， $d = 1$ ，如图 5-33 所示，然后按[OK]键确认。

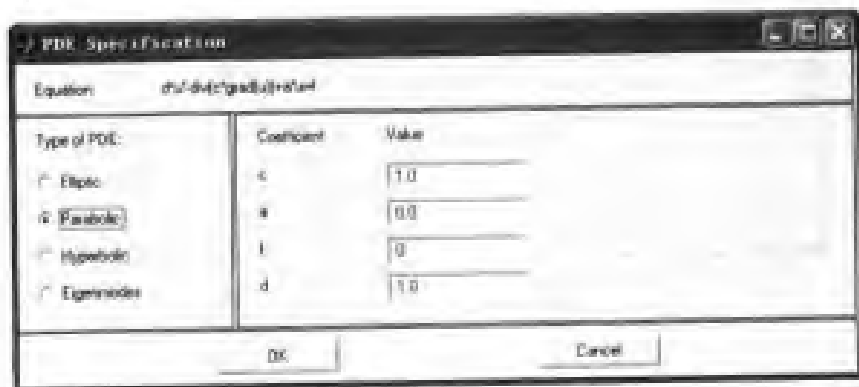


图 5-33 PDE 方程的系数设置

(5) 网格初始化及精细化

单击初始化网格，再单击精细化网格，然后单击 Jiggle，得到图 5-34。

(6) 设置初值、求解 PDE

执行[Solve]->[Parameters...], 弹出如图 5-35 的求解参数设置对话框，设置后按[OK]键确认。然后按工具栏中的“=”求解 PDE，得到解的图形（图 5-36）。最后保存为 *PDEtool2DD_CrackHT.m*。

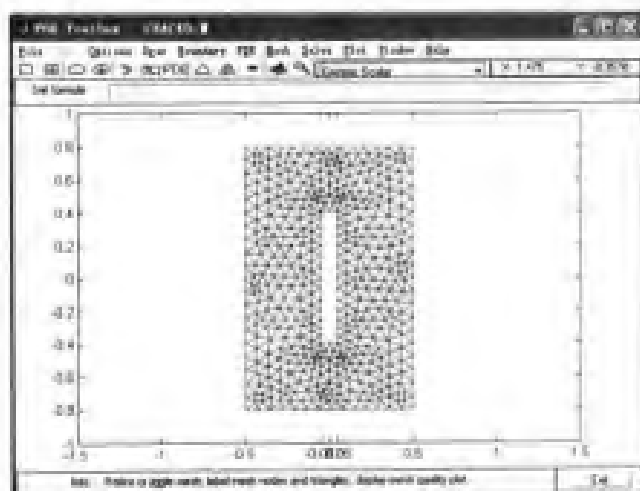


图 5-34 精细化后的网格



图 5-35 求解参数的设置

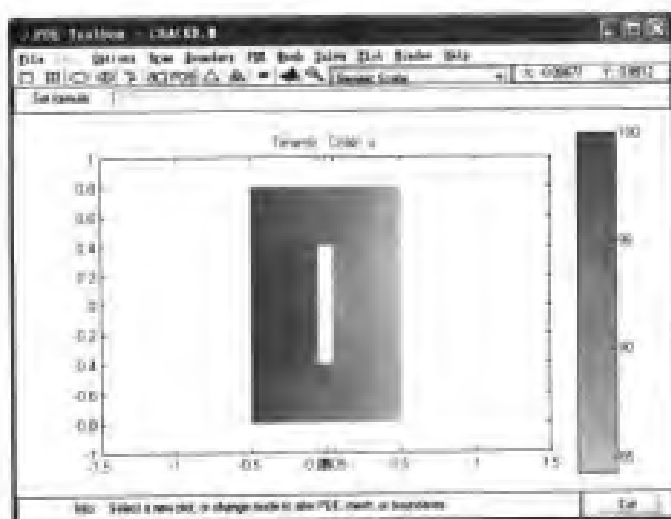


图 5-36 问题的解

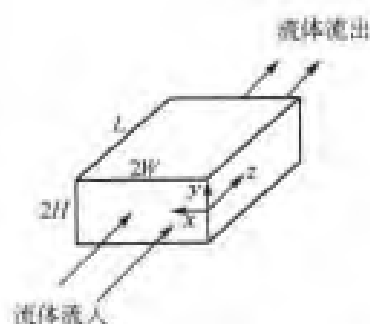


图 5-37 矩形管道中的流体流动

【例 5-11】 用 pde tool 模拟计算矩形管道中的牛顿型流体流动（二维动态 PDE 模型）^[4]

如图 5-37 所示，一牛顿型流体在长 L 、宽 $2W$ 、高 $2H$ 的矩形管道中流动。开始时 ($t=0$)，流体处于静止状态，在压力梯度的强制作用下流体开始流动。假设流体密度和粘度为常数，动量衡算方程为

$$\rho \frac{du}{dt} = \frac{p_0 - p_L}{L} + \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (a)$$

式中, ρ 为密度, $\frac{p_0 - p_L}{L}$ 为压力降; μ 为流体黏度; u 为轴向流体速度。

试研究流体在管道中的速度分布。

解 先进行无因次化, 即设 $X = \frac{x}{H}$, $Y = \frac{y}{H}$, $U = \frac{u}{(p_0 - p_L)H^2/(2\mu L)}$, $\tau = \frac{\mu t}{\rho H^2}$, 代入 (a) 得

$$\frac{\partial U}{\partial \tau} = 2 + \frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2} \quad (\text{b})$$

$$\text{初始条件:} \quad \tau = 0, U = 0 \quad (\text{c})$$

边界条件 (考虑对称性, X 、 Y 求解域取整个矩形界面的四分之一):

$$X = 0, U = 0 \quad (\text{壁上}) \quad (\text{d})$$

$$Y = 1, U = 0 \quad (\text{壁上}) \quad (\text{e})$$

$$X = \frac{W}{H}, \frac{\partial U}{\partial X} = 0 \quad (\text{对称性}) \quad (\text{f})$$

$$Y = 0, \frac{\partial U}{\partial Y} = 0 \quad (\text{对称性}) \quad (\text{g})$$

式 (b) ~ (g) 是抛物型 PDE 初始-边值问题, 对照标准抛物型 PDE 方程 (5-29a) 可知, 方程的系数为: $d = 1$, $c = 1$, $a = 0$, $f = 2$ 。这样, 可直接用 `pdetool` 求解, 程序名为 `PDEtool2DD_FluidFlow.m`。计算得到的流体速度分布如图 5-38 所示。

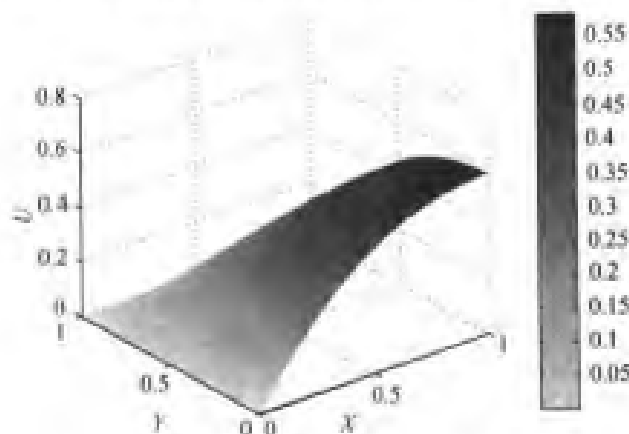


图 5-38 流体在管中的速度分布

注 按一般的无因次化方法, 通常 X 应设为 $X = \frac{x}{B}$, 但这时导出的无因次化方程为:

$$\frac{\partial U}{\partial \tau} = 2 + \left(\frac{B}{W}\right)^2 \frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2}$$

显然, 此方程形式上与标准抛物型 PDE 方程 (5-29a) 不一致, 不能直接用 `pdetool` 或命令行函数求解, 所以为便于计算, X 的无因次化应按式 (b) 进行。

5.5.4 使用 PDE 工具箱的命令行函数求解偏微分方程

在 5.5.3 介绍如何用 `pdetool` 求解 PDE 问题, 该方法运行 `pdetool` 后, 只需通过其界面定义 PDE 方程 (包括方程类型、方程系数), 初始条件和边界条件等, 即可求出方程的解, 它是一种不用编程的傻瓜式方法, 使用很方便。但是, 待解的 PDE 问题必须完全符合 `pdetool`

所能求解的 PDE 问题的标准形式(包括方程类型、初始条件和边界条件等)。此外,用 `pdetool` 图形用户界面只能求解方程数为 2 以内的 PDE 标准问题,3 个方程以上的 PDE 问题,即使符合标准形式,也无法用它求解。这时可以考虑使用命令行函数(Command-Line Functions)的方法。

为便于掌握通过编程使用命令行函数的求解方法,下面先介绍用有限元(FEM)法求解 PDE 方程的过程,接着介绍使用 MATLAB 命令行函数计算 PDE 方程的一般步骤、常用 PDE 函数的功能及其使用方法,然后举例。

用有限元法求解 PDE 方程的过程

有限元法本身是一种离散方法,它将一个连续体剖分为“有限个基本元”,然后用有限个参数描述这个基本元上的物理特性,建立平衡方程,把连续问题转化为离散问题。在有限个点上求出解的近似值。

以椭圆型 PDE 问题(5-28a)、(5-33)~(5-35)为例,用有限元法求解 PDE 方程的过程如下。

① 剖分区域,将区域分为有限个互不重叠的三角形“基本元”,根据实际问题的需要在求解域 Ω 内取 N_p 个点,并使这些点连成三角形网格,每个三角形就是一个“基本元”,三角形基本元的顶点称为“节点”,有一条落在边界的基本元称为“边界基本元”,其余称为“内部基本元”。例如,在一个矩形区域内生成的三角形网格如图 5-39(a)所示;

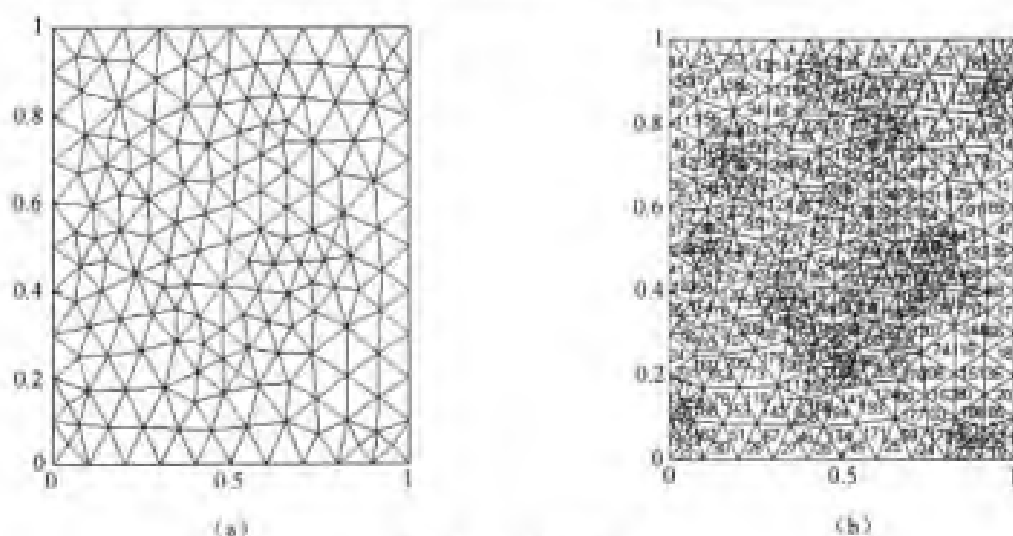


图 5-39 三角形网格

② 将基本元编号,如图 5-39(b);

③ 在每个基本元(包括边界基本元在内)上构造变分问题解的线性插值函数 ϕ_i (i 为基本元编号);

④ 将每一个单元上构造的函数 ϕ_i 合并起来,就得到在整个区域 Ω 上的分块近似函数。并且定义

$$K_{i,j} = \int_{\Omega} (c \nabla \phi_i) \cdot \nabla \phi_j dx$$

$$M_{i,j} = \int_{\Omega} a \phi_i \phi_j dx$$

$$Q_{i,j} = \int_{\Gamma} q \phi_i \phi_j ds$$

$$F_i = \int_G f_i \phi_i dx$$

$$G_i = \int_T g \phi_i ds$$

⑤ 总体合成

将 K_{ij} 、 M_{ij} 、 Q_{ij} 、 F_{ij} 、 G_{ij} 合成矩阵 K 、 M 、 Q 、 F 、 G (其中, K 称为刚度矩阵, M 称为质量矩阵, Q 为边界条件 q 的贡献, F 为右端向量, G 为边界条件 g 的贡献), 则由变分原理及有限元的理论可知, 原微分方程等价于线性方程

$$(K + M + Q)U = F + G \quad (5-47)$$

其中 K 、 M 、 Q 是 $N_p \times N_p$ 矩阵, F 、 G 是 N_p 维向量。

可以用 MATLAB 函数 `assemma()` 计算 K 、 M 、 F , 再用 `assemb()` 计算 Q 、 G , 则 PDE 的解为:

$$U = (K + M + Q)^{-1}(F + G)$$

也可用 MATLAB 左除运算符“\”求出 PDE 的解:

$$U = (K + M + Q) \setminus (F + G) \quad (5-48)$$

用 PDE 函数计算 PDE 方程的一般步骤及相应的 MATLAB 函数。

- ① 问题定义及参数初始化——`pdetool()`、`wbound()`、`wgeom()`
定义求解域 (几何条件)、边界条件和方程系数及参数初始化;
- ② 网格化 (三角形网格划分、网格细化及微调)——`initmesh()`、`refinemesh()`、`jigglemesh()`
为便于观察网格, 还可用函数 `pdemesh(p, e, t)` 绘制由网格数据 p, e, t 指定的网格;
- ③ 求解 PDE——`assemblpde()`、`parabolic()`、`hyperbolic()`、`pdenonlin()`;
- ④ 显示结果 (包括绘图、动画等): `pdesurf()`、`pdeplot()`、`colormap()`。

PDE 工具箱的常用函数

MATLAB PDE 1.0.4 (R13) 版共有 50 多个函数, 下面仅介绍最常用的 20 个函数。

(1) 求解 PDE 的函数 (PDE 求解器)

PDE 工具箱求解 PDE 的函数 (PDE 求解器) 如表 5-7 所示。

表 5-7 PDE 求解函数列表

函 数	PDE 方程	功 能
<code>adaptmesh</code>	(5-28a) 或 (5-28b)	生成自适应网格并求解 PDE
<code>assemma</code>		集合面积分的贡献
<code>assemb</code>		集合边界条件的贡献
<code>assemblpde</code>	(5-28a) 或 (5-28b)	组成 PDE 方程的刚度矩阵及右边项 (椭圆型)
<code>hyperbolic</code>	(5-30a) 或 (5-30b)	求解双曲型 PDE 方程
<code>parabolic</code>	(5-29a) 或 (5-29b)	求解抛物线型 PDE 方程
<code>pdeeig</code>	(5-31a) 或 (5-31b)	求解特征值 PDE 方程
<code>pdenonlin</code>	(5-28c) 或 (5-28d)	求解非线性椭圆型 PDE 方程
<code>poisolv</code>		在矩形网格上快速求解泊松方程

函数 `assemblpde()`

功能: 为 PDE 问题 (5-28a) 或 (5-28b) 组合刚度矩阵和右端向量。函数 `assemblpde()` 是 PDE 工具箱中的一个基本函数, 它使用有限元法来组合与求解 PDE 问题 (5-28a) 或 (5-28b)。

调用格式为:


```

u = assempde(b, p, e, t, c, a, f)
[K, F] = assempde(b, p, e, t, c, a, f)
[K, F] = assempde(b, p, e, t, c, a, f, u0, time, sdl)
[K, F] = assempde(b, p, e, t, c, a, f, time, sdl)
[K, F, B, ud] = assempde(b, p, e, t, c, a, f)
[K, M, F, Q, G, H, R] = assempde(b, p, e, t, c, a, f)
[K, M, F, Q, G, H, R] = assempde(b, p, e, t, c, a, f, u0, time, sdl)
[K, M, F, Q, G, H, R] = assempde(b, p, e, t, c, a, f, time, sdl)
u = assempde(K, M, F, Q, G, H, R)

```

输入参数:

- b** 描述 PDE 问题的边界条件, 可以是矩阵或 M 文件。边界条件矩阵和边界 M 文件的格式可分别参考函数 `assemb()` 和 `pdebound()`。
- p, e, t** 描述 PDE 问题几何模型的网格数据。有关网格数据的细节可参考函数 `initmesh()` 或 `pdegeom()`。
- u0, time** 可选输入变量 `u0` 和 `time` 分别用于非线性求解器和时间步进算法 (time stepping algorithms)。初值 `u0` 的形式与 `u` 相同。
- sdl** 使用可选输入变量 `sdl` (子域标签) 时, 只对 `sdl` 标签所标示的子域进行组合。

输出参数:

- u** PDE 问题的解向量
- K, M, F** 分别是刚度矩阵、质量矩阵和右边向量, 参考式 (5-47): $(K + M + Q)U = F + G$
- Q, G** 分别表示混合边界条件式 (5-35) 中 q 和 g 的贡献
- H, R** 方程 $H \cdot u = R$ 代表 Dirichlet 边界条件, 即 (5-33) 式。

说明:

- `u = assempde(b, p, e, t, c, a, f)` % 通过从线性方程组消除 Dirichlet 边界条件来组合和求解 PDE 问题
- `[K, F] = assempde(b, p, e, t, c, a, f)` % 通过用刚度矩阵 K 和 F 近似 Dirichlet 边界条件来组合和求解 PDE 问题, 方程解为 $u = K \setminus F$
- `[K, F, B, ud] = assempde(b, p, e, t, c, a, f)` % 通过从线性方程组消除 Dirichlet 边界条件来组合 PDE 问题。非 Dirichlet 点上的解为 $un = K \setminus F$, 完整 PDE 问题的解为 $u = B \cdot un + ud$ 。
- `[K, M, F, Q, G, H, R] = assempde(b, p, e, t, c, a, f)` % 将 PDE 问题分解为 K, M, F, Q, G, H, R
- `u = assempde(K, M, F, Q, G, H, R)` % 将分解后的 K, M, F, Q, G, H, R 转化为单一矩阵或向量的形式, 然后通过从线性方程组中消除 Dirichlet 边界条件求得方程解 u 。
- `[K1, F1] = assempde(K, M, F, Q, G, H, R)` 通过用很大的刚度常数修改边界条件将分解后的 K, M, F, Q, G, H, R 转化为单一矩阵或向量的形式。
- `[K1, F1, b, ud] = assempde(K, M, F, Q, G, H, R)` 通过从线性方程组中消除 Dirichlet 边界条件将分解后的 K, M, F, Q, G, H, R 转化为单一矩阵或向量的形式。

对于标量 PDE 的情况, 其解 u 是一个列向量, 列向量中的元素即是 p 的对应节点上的值。

对于有 N_p 个节点的 N 维系统 (PDE 方程组), u 的最前面 np 个值即是 N_p 个节点上的 u_1 分量值, 接下来的 N_p 个值即是 N_p 个节点上的 u_2 分量值, 如此类推。这样, u 的元素 (u_1, u_2, \dots, u_N) 就作为 N 个节点值块放置到向量 u 中。

标量形式的 PDE 方程系数 c, a 和 f 的表示

- 常数值;
- 由三角形基本元的中心点上的取值构成一个行向量;
- 由 MATLAB 函数及程序产生。

PDE 系统 (方程组) 问题的系数

对于 N 维 PDE 方程组, c 是一个 $N \times N \times 2 \times 2$ 的张量 (4 维数组), a 是一个 $N \times N$ 的矩阵, f 是一个长度为 N 的列向量。c、a 和 f 的元素 c_{ijkl} 、 a_{ij} 和 f_i 按照行的大小保存在 MATLAB 矩阵 c、a 和 f 中 (详细参考函数 assempde 的帮助功能)。

函数 assema()

功 能: 在 pde 问题中集合面积分的贡献

语 法: $[K, M, F1] = \text{assema}(p, t, c, a, f)$

$[K, M, F1] = \text{assema}(p, t, c, a, f, u0, \text{time}, \text{sdl})$

$[K, M, F1] = \text{assema}(p, t, c, a, f, \text{time}, \text{sdl})$

输入参数: p, t, c, a, f, u0, time, sdl 同 assempde()。

输出参数: K, M, F1 分别是刚度矩阵、质量矩阵和右边向量 F1。

函数 assemb()

功 能: 在 pde 问题中集合边界条件的贡献

语 法: $[Q, G, h, r] = \text{assemb}(b, p, e)$

$[Q, G, h, r] = \text{assemb}(b, p, e, \text{time}, \text{sdl})$

$[Q, G, b, r] = \text{assemb}(b, p, e, u0, \text{time}, \text{sdl})$

输入参数: b, p, e, u0, time 和 sdl 同 assempde()。

输出参数: Q, G 分别表示混合边界条件式 (5-35) 中 q 和 g 的贡献,

H, R 方程 $H \cdot u = R$ 代表 Dirichlet 边界条件, 即 (5-33) 式。

函数 Parabolic()

功能: 求解抛物线型 PDE 问题。

调用格式: $u1 = \text{parabolic}(u0, \text{tlist}, b, p, e, t, c, a, f, d)$

研究域为由 p, e 和 t 描述的网格, b 为边界条件, 它可以是一个边界条件矩阵或边界 M 文件的文件名。边界条件可以与时间 t 有关。初值为 u0。系数 c, a, d 和 f 也可以与时间 t 有关, 它们可以通过多种方法给定。参见 assempde() 的帮助。

对于标量的情况, 解矩阵 u1 中的每一行都是由 p 中对应列给定的坐标上的解。u1 中的每一列为 tlist 中对应项给定的时间上的解。对于具有 np 个节点的 N 维系统, u1 的前 np 列描述 u 的第一个元素, 接下来的 np 行描述 u 的第二个元素, 如此类推。这样, u 的元素就作为节点行的 N 个块放到 u 中。

$u1 = \text{parabolic}(u0, \text{tlist}, K, F, B, ud, M)$ 用于求解 ODE 方程问题:

$$B' M B \frac{du_i}{dt} + K \cdot u_i = F$$

$$u = B u_i + u_d$$

u 的初值为 u0。

函数 hyperbolic()

功能: 求解双曲线 PDE 问题。

调用格式: $u1 = \text{hyperbolic}(u0, ut0, tlist, b, p, e, t, c, a, f, d)$

参数: $ut0$ 为初始导数, 其他参数与 $\text{Parabolic}()$ 完全相似。函数 hyperbolic 用法同 $\text{Parabolic}()$ 。

$u1 = \text{hyperbolic}(u0, ut0, tlist, k, f, b, ud, m)$ 用于求解 ODE 方程问题:

$$B'MB \frac{d^2 u_i}{dt^2} + K \cdot u_i = F$$

$$u = Bu_i + u_d$$

u 的初值为 $u0$, 初始导数值为 $ut0$ 。

adaptmesh()

功 能: 生成自适应网格并求解 PDE 方程

语 法: $[u, p, e, t] = \text{adaptmesh}(g, b, c, a, f)$.

$[u, p, e, t] = \text{adaptmesh}(g, b, c, a, f, [p1, v1, \dots])$.

返回值: u 解向量 (列向量), 详见 assemblpde 。

p, e, t 即描述 PDE 问题的自适应三角形网格数据, 详见 initmesh 或 pdegeom 。

参 数: g 描述 PDE 问题的几何条件。 g 可以是分解几何矩阵或几何 M 文件的文件名。

b 描述 PDE 问题的边界条件。 b 可以是边界条件矩阵或边界 M 文件的文件名。

c, a, f 即 PDE 方程的系数。 c, a, f 可通过多种方式给出, 详见 assemblpde 。

注 释: 该函数生成椭圆型标量 pde 问题

$$-\nabla \cdot (c \nabla u) + au = f$$

的解, 其几何条件和边界条件由 g 和 b 给出, 网格由 p, e, t 描述。

Par 传递给 tripick 函数。它通常作为衡量解与方程拟合程度的容许误差, 当生成三角形的最大次数达到 ngen 或网格中的三角形个数超过 maxt 时, 网格精细化结束。

$p1, e1$ 和 $t1$ 为输入网格数据, 该三角形网格可用作自适应算法的初始网格, 如果尚未提供初始网格, 则可通过调用无参数的 initmesh 函数得到初始网格。

三角形选择方法是一种用户定义的三角形选择方法。由函数 pdejumps 计算得到误差估计值后, 选择在下一步网格生成中将要细化的三角形。使用参数 $p, t, cc, aa, ff, u, \text{errf}$ 和 par 调用函数。 p 和 t 代表当前生成的三角形, cc, aa 和 ff 是 pde 问题的当前系数 (已扩展到三角形的中点), 见表 5-8。

表 5-8 有效属性及属性值

属 性 名	属 性 值	缺 省 值	属 性 描 述
Maxt	positive integer	inf	新三角形的最大个数
Ngen	positive integer	10	生成三角形的最大次数
Mesh	p1, e1, t1		初始网格
Tripick	{pdeadworst} pdeadgsc		三角形选择方法
Par	numeric	0.5	函数参数
Rmethod	{longest} regular		三角形精细化方法
Nonlin	on : off		使用非线性求解器
Toln	numeric	1e-3	非线性容许误差
Init	string numeric		非线性初值
Jac	{fixed} lumped full		非线性雅可比矩阵的计算
Norm	numeric	inf	非线性残差范数

u 是当前解, $errf$ 是计算得到的误差估计值, 而 par 是作为选项传递给 `adaptmesh` 的参数。矩阵 cc , aa , ff 和 $errf$ 都有 nt 列, 其中 nt 为当前三角形的个数。矩阵 cc , aa 和 ff 的行数正好与输入变量 c , a 和 f 相同。 $errf$ 中的每一行对应系统中某一个方程。`pde` 工具箱中有两个标准的三角形选择方法, 即 `pdeadworst` 和 `pdeadgsc`。

函数 `pdenonlin()`

功 能: 求解非线性椭圆型偏微分方程。

调用格式: `[u, res] = pdenonlin(b, p, e, t, c, a, f)`

输入参数: b 是边界条件矩阵或边界 M 文件的文件名, p, e, t 为网格数据, c, a, f 为方程系数。

输出参数: u 为解向量 (列向量), res 为牛顿步长残差。

(2) 区域划分及有限元网格描述 (几何算法) 函数

这类函数有: `initmesh`、`jigglemesh`、`refinemesh`、`wbound`、`wgeom`、`pdebound`、`pdegeom`。

函数 `initmesh()`

功 能: 创建初始的三角形网格。

调用格式: `[p, e, t] = initmesh(g)`

输入参数: g 描述 PDE 问题的几何条件。 g 可以是分解几何矩阵或几何 M 文件的文件名。

输出参数: p, e, t 为网格数据 (同前)。

函数 `refinemesh()`

功 能: 精化三角形网格。

调用格式: `[p1, e1, t1] = refinemesh(g, p, e, t)`

参数说明: 输入参数 g, p, e, t 同前, 输出参数 $p1, e1, t1$ 为区域 g 上精化后的网格数据。

函数 `jigglemesh()`

功 能: 微调三角形网格内部点, 以提高网格的质量。

调用格式: `p1 = jigglemesh(p, e, t)`

参数说明: 输入参数 p, e, t 同前, 输出参数 $p1$ 为区域 g 上微调后的网格数据。

函数 `wbound()`

功 能: 把边界条件的有关数据 (bl) 保存为边界条件 M 文件。

调用格式: `fid = wbound(bl, 'MN')`

输入参数: bl 为边界条件矩阵, 可用 `pdetool` 获得; MN 为边界 M 文件名, 扩展名自动取 `.m`;

输出参数: 若不能写入边界 M 文件 $MN.m$, 则 fid 返回 -1。

函数 `wgeom()`

功 能: 把几何模型的有关数据 (g) 保存为几何模型 M 文件。

调用格式: `fid = wgeom(dl, 'MN')`

输入参数: dl 为分解几何矩阵, 可用 `pdetool` 获得; MN 为几何 M 文件名, 扩展名为 `.m`;

输出参数: 若不能写入文件时, fid 返回 -1。

函数 `pdegeom()` 和函数 `pdebound()`

函数 `pdegeom()` 和 `pdebound()` 分别用于建立几何 M 文件和生成边界条件 M 文件。但此项工作可用 `pdetool` 来完成, 此方法比较简便, 因此 `pdegeom()` 和 `pdebound()` 的用法不再作详细

介绍。

(3) 画图函数 常用的画图函数有: `pdegplot`、`pdemesh`、`pdeplot`、`pdesurf`。

函数 `pdeplot()`

功 能: 普通的 PDE 工具箱绘图函数, 它可以同时显示几个函数的 PDE 解。

调用格式: `pdeplot(p, e, t, pl, vl, ...)`

输入参数: `p, e, t` 同前, `pl, vl` 分别是属性名和属性值。

函数 `pdegplot()`

功 能: 绘制 PDE 几何图形。

调用格式: `pdegplot(g)`

输入参数: `g` 可以是分解几何矩阵或几何 M 文件的文件名 (同前)。

函数 `pdemesh()`

功 能: 绘制 PDE 三角形网格, 用此函数绘图的速度要比 `pdesurf()` 快。

调用格式: `pdemesh(p, e, t)` % 绘制由网格数据 `p, e, t` 定义的网格

`pdemesh(p, e, t, u)` % 使用网格图绘制解的列向量 `u`。

输入参数: `p, e, t` 同前, `u` 为解向量 (列向量)。

函数 `pdesurf()`

功 能: 绘制 PDE 解的三维表面图。

调用格式: `pdesurf(p, t, u)`

输入参数: `p, t, u` 同前。

(4) 工具算法 (Utility algorithms) —— `tri2grid()`

函数 `tri2grid()`

功 能: 将由 `p, t` 描述的三角形网格上的函数 `u` 转换为由向量 `x` 和 `y` 描述的矩形网格上的函数 `uxy`。

调用格式: `uxy = tri2grid(p, t, u, x, y)`

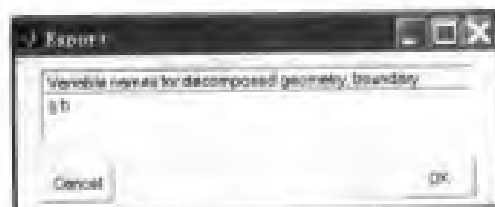
【例 5-12】 利用 PDE 工具箱函数求解[例 5-9]的泊松方程。

解 (1) 问题定义及参数初始化

这一步骤包括定义求解域的几何条件、边界条件、方程系数及参数初始化等。其中, 求解域几何条件和边界条件 M 文件可用 `pdetool` 获得, 这是最简便的方法。

由于例 5-9 已用 `pdetool` 求解此泊松方程, 并保存为 `PDEtoolPoisson.m`, 因此, 这里可直接打开此文件以生成求解域几何条件和边界条件 M 文件, 具体操作如下:

打开 `PDEtoolPoisson.m`, 执行 `[Boundary]->[Export Decomposed Geometry, Boundary Cond's...]`, 即弹出下图:



单击[OK]按钮, 则输出分解几何变量 `g` 和边界变量 `b` 到 MATLAB 工作环境中。这时, 可切换到 MATLAB 工作环境, 显示 `g` 和 `b` 观察其内容, 再执行以下两条命令把边界条件和求解域 (几何条件) 存入文件 `poissonb.m` 和 `poissong.m` 中, 以供程序调用。

```
>>wbound(b, 'poissonb')
```

```
>>wgeom(g, 'poissong')
```

这样, 实现问题定义及参数初始化这一步骤的代码如下:

```
g = 'poissong';    % 定义求解域  
b = 'poissonb';    % 定义边界条件  
c = 1.0;  a = 0.0;  f = 2;
```

(2) 网格化 (三角形网格划分及网格细化)

```
[p, e, t] = initmesh(g);    % 网格初始化  
[p, e, t] = refinemesh(g);  % 网格细化  
pdemesh(p, e, t);          % 绘制由网格数据 p, e, t 指定的网格
```

(3) 求解 PDE

```
u = assempde(b, p, e, t, c, a, f)
```

(4) 显示结果 (包括绘图、动画等)

```
pdesurf(p, t, u)  
pdeplot(p, e, t)  
colormap(cool)
```

程序清单 见光盘中的 *Poisson.m*。

【例 5-13】 求解二维动态热传导问题^[5]

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}, \quad (0 < x < 15, 0 < y < 20, t > 0)$$

I.C. $u|_{t=0} = 0, \quad (0 < x < 15, 0 < y < 20)$

B.C. $u|_{x=0} = u|_{x=15} = u|_{y=0} = u|_{y=20} = 100, \quad (t \geq 0)$

程序说明 这是抛物型 PDE 问题, 可用 *parabolic()* 求解。程序中, 几何模型 M 文件 *heatcondg.m* 和边界条件 M 文件 *heatcondb.m* 是用 *pde tool* 按 5.5.3.1 的求解步骤 (1) ~ (3) 得到的, 即: 运行 *pde tool* 后, 用 Options 菜单设置应用模式为 Heat Transfer, 再利用 Draw 绘制几何模型, 设置矩形求解域为: [0, 15], [0, 20], 然后定义边界条件, 并执行 Boundary 菜单的 Export Decomposed Geometry, Boundary cond 功能把几何模型和边界条件数据 (g 和 b) 输出到 MATLAB 工作环境, 最后切换到工作环境并执行下面命令:

```
>>wgeom(g, 'heatcondg')  
>>wbound(b, 'heatcondb')
```

程序清单 见光盘中 *HeatCond_2DD.m*。

计算结果 30 秒时的温度分布示于图 5-40 中。

【例 5-14】 利用 PDE 工具箱函数通过编程模拟计算矩形管道中的牛顿型流体流动 (问题同[例 5-11])

程序说明 这是抛物型 PDE 问题, 可用函数 *parabolic()* 求解。主程序为 *PDE2DD_FluidFlow.m*, 其中, 几何模型 M 文件 *FluidFlowg.m*

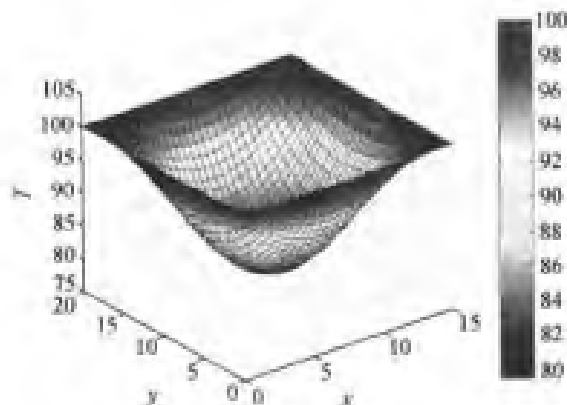


图 5-40 30 s 时的温度分布

和边界条件 M 文件 FluidFlowb.m 由 pdetool 得到。

程序清单 PDE2DD_FluidFlow.m

```
function PDE2DD_FluidFlow          % 命令方式计算, 问题同 PDE2DDtool_FluidFlow.m
clear all; clc
% 1. 问题定义及参数初始化
g = 'FluidFlowg';          % 定义求解域
b = 'FluidFlowb';          % 定义边界条件
c = 1.0; a = 0.0; d = 1.0; f = 2;      % 方程的系数
% 2. 网格化 (三角形网格划分及网格细化)
[p, e, t] = initmesh(g);          % 网格初始化
[p, e, t] = refinemesh(g, p, e, t); % 网格细化
% 3. 绘制 PDE 三角形网格
pdemesh(p, e, t);
% 4. 给定初值条件并求解 PDE
u0 = 0; tlist = [0:0.1:1];
u1 = parabolic(u0, tlist, b, p, e, t, c, a, f, d)
% 5. 显示结果 (包括绘图)
x = [0:0.01:1]; y = [0:0.01:1];          % x = 0~1, y = 0~1
uxy = tri2grid(p, t, u1(:, find(tlist==1)), x, y) % tau = 1
figure, surf(x, y, uxy), xlabel('x'), ylabel('y'), zlabel('U')
```

计算结果 流体在管道中的速度分布如图 5-41 所示, 可见该结果与例 5-11 的结果基本相同。

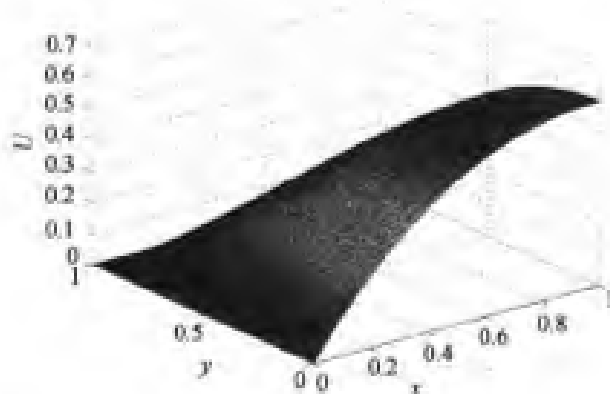


图 5-41 流体在管道中的速度分布

【例 5-15】 计算等温一级反应的圆柱状催化剂颗粒中的浓度分布和有效因子^[3]

数学模型 在短圆柱状催化剂颗粒中进行等温一级反应, 圆柱的直径为 D , 长为 L 。假设坐标原点在圆柱的中心, 则代表短圆柱内 A 的浓度分布的模型方程为:

$$D_A \left[\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial C_A}{\partial r} \right) + \frac{\partial^2 C_A}{\partial z^2} \right] = k C_A \quad (1)$$

其中扩散系数 D_A 和反应速率常数 k 为定值。

$$\text{边界条件: } z = \pm \frac{L}{2}, \quad 0 \leq r \leq \frac{D}{2}, \quad C_A = C_{A0} \quad (2)$$

$$r = 0, \quad -\frac{L}{2} \leq z \leq \frac{L}{2}, \quad \frac{\partial C_A}{\partial r} = 0 \quad (3)$$

$$r = \frac{D}{2}, \quad -\frac{L}{2} \leq z \leq \frac{L}{2}, \quad C_A = C_{A0} \quad (4)$$

$$\text{催化剂的内部有效因子为: } \eta = \frac{\iiint_V R_A dV}{\iiint_V R_{A0} dV} \quad (5)$$

式中, V 为催化剂颗粒体积; R_A 为单位体积催化剂颗粒中生成 A 的局部速率。

将以上模型无因次化: 令 $C = \frac{C_A}{C_{A0}}$, $R = \frac{2r}{D}$, $Z = \frac{2z}{L}$, $\phi = \frac{D}{2} \sqrt{\frac{k}{D_A}}$, 则式 (1) ~ 式 (5) 变成

$$\frac{1}{R} \frac{\partial}{\partial R} \left(R \frac{\partial C}{\partial R} \right) + \left(\frac{D}{L} \right)^2 \frac{\partial^2 C}{\partial Z^2} = \phi^2 C \quad (6)$$

$$R = 0, \quad -1 \leq Z \leq 1, \quad \frac{\partial C}{\partial R} = 0 \quad (7)$$

$$R = 1, \quad -1 \leq Z \leq 1, \quad C = 1 \quad (8)$$

$$Z = \pm 1, \quad 0 \leq R \leq 1, \quad C = 1 \quad (9)$$

$$\eta = \frac{\int_0^1 \int_{-1}^1 CR dR dZ}{\int_0^1 \int_{-1}^1 R dR dZ} \quad (10)$$

已知: $\phi = 4$, $L = 1$, $D = 0.5$ 。

程序说明 容易看出, 方程 (6) 是椭圆型 PDE 方程, 但它不是形如 (5-28a) 式的标准形式, 为了使用 MATLAB PDE 工具箱, 必须先将它变换为 (5-28a) 式的标准形式, 即令

$$x \equiv R, \quad y \equiv \frac{L}{D} Z, \quad u \equiv C$$

$$\text{则式 (6) 变为: } \frac{\partial}{\partial x} \left(x \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(x \frac{\partial u}{\partial y} \right) = x \phi^2 u$$

即

$$-\nabla \cdot x \nabla u = -x \phi^2 u \quad (11)$$

相应地, 边界条件式 (7) ~ (9) 变为:

$$x = 0, \quad -\frac{L}{D} \leq y \leq \frac{L}{D}, \quad \frac{\partial u}{\partial x} = 0 \quad (12)$$

$$x = 1, \quad -\frac{L}{D} \leq y \leq \frac{L}{D}, \quad u = 1 \quad (13)$$

$$y = \pm \frac{L}{D}, \quad 0 \leq x \leq 1, \quad u = 1 \quad (14)$$

式 (10) 变为

$$\eta = \frac{\int_{-0.5}^{0.5} \int_0^1 u x dx dy}{\int_{-0.5}^{0.5} \int_0^1 x dx dy} \quad (15)$$

将式 (11) 与标准形式 (5-28a) 对照, 可得方程系数为:

$$c = x, a = 0, f = -x\phi^2 u \quad (16)$$

式(11)~式(14)是标准的非线性椭圆型 PDE 边值问题, 程序用函数 `pdenonlin()` 求解, 其中几何 M 文件 `CylindCat1g.m` ($L/D = 0.5$) 和边界条件 M 文件 `CylindCat1b.m` 同前所述由 `pdetool` 得到。对于式(15), 容易算得其分母的值为 $1/2$, 而分子用双重积分函数 `dblquad()` 计算。

注 由于 c 、 f 与 x 、 u 有关, 程序中 c 、 f 的表达式要用字符串表示。

程序清单 *CylindCat1.m*

```
function CylindCat1
clear all; clc
% 1. 问题定义及参数初始化
g = 'CylindCat1g';      % 定义求解域
b = 'CylindCat1b';      % 定义边界条件
c = 'x'; a = 0; f = '-4*x.*u'; % 方程的系数

% 2. 网格化 (三角形网格划分及网格细化)
[p, e, t] = initmesh(g); % 网格初始化
[p, e, t] = refinemesh(g, p, e, t); % 网格细化
p = jigglemesh(p, e, t);
% 3. 绘制 PDE 三角形网格
pdemesh(p, e, t);

% 4. 求解 PDE
[u, res] = pdenonlin(b, p, e, t, c, a, f)
% 求解圆柱形催化剂内部有效因子 eta
xi = linspace(0, 1, 50); yi = linspace(-0.5, 0.5, 50);
for i=1:length(xi)
    for j=1:length(yi)
        u1 = tri2grid(p, t, u, xi(i), yi(j));
        uxy(i, j) = u1;
    end
end
I = dblquad(@Func, 0, 1, -0.5, 0.5, [], @quadl, xi, yi, uxy);
eta = 2*I
% 5. 显示结果 (包括绘图、动画等)
figure, pdesurf(p, t, u), colorbar
xlabel('x'); ylabel('y'); zlabel('u')
% -----
function f = Func(x, y, xi, yi, ui)
u = interp2(xi, yi, ui, x, y);
f = u.*x;
```

计算结果 $\eta = 0.8371$ 。

【例 5-16】 计算圆柱状催化剂颗粒中进行一级反应的浓度分布^[4]

数学模型 圆柱形催化剂颗粒长径比为 L/D ，在催化剂中进行一级反应，其传递方程为：

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial c}{\partial r} \right) + \left(\frac{D}{L} \right)^2 \frac{\partial^2 c}{\partial z^2} = \phi^2 c \exp\left(\frac{\gamma}{T}(T-1)\right) \quad (1)$$

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T}{\partial r} \right) + \left(\frac{D}{L} \right)^2 \frac{\partial^2 T}{\partial z^2} = -\beta \phi^2 c \exp\left(\frac{\gamma}{T}(T-1)\right) \quad (2)$$

式中， r 为量纲一的径向坐标 ($0 \leq r \leq 1$)； z 为量纲一的轴向坐标 ($0 \leq z \leq 1$)； C 为量纲一的浓度； T 为量纲一的温度； r 为量纲一的 Arrhenius； β 为量纲一的 Prater 数。

$$\text{B.C.} \quad r=0, \quad \frac{\partial c}{\partial r} = \frac{\partial T}{\partial r} = 0 \quad (3)$$

$$z=0, \quad \frac{\partial c}{\partial z} = \frac{\partial T}{\partial z} = 0 \quad (4)$$

$$r=1, \quad z=1, \quad c=T=1 \quad (5)$$

$$\text{利用 Prater 关系式:} \quad T=1+(1-c)\beta \quad (6)$$

则 PDE 方程组的边值问题 (1) ~ (5) 变成单个 PDE 方程的边值问题：

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial c}{\partial r} \right) + \left(\frac{D}{L} \right)^2 \frac{\partial^2 c}{\partial z^2} = \phi^2 c \exp\left[\frac{\gamma\beta(1-c)}{1+\beta(1-c)}\right] \quad (7)$$

$$r=0, \quad \frac{\partial c}{\partial r} = 0 \quad (8)$$

$$z=0, \quad \frac{\partial c}{\partial z} = 0 \quad (9)$$

$$r=1, \quad z=1, \quad c=1 \quad (10)$$

已知参数： $\beta=0$ ， $r=30$ ， $\phi=3$ ， $L/D=1$ 。

试计算在位置 $(r, z) = (0.394, 0.285)$ 、 $(0.394, 0.765)$ 、 $(0.803, 0.285)$ 、 $(0.803, 0.765)$ 上的浓度。

程序说明 方程 (7) 与上例的方程 (6) 都是椭圆型 PDE 方程，仅仅是方程右边项的表达式及问题的求解域略有不同，解法是完全一样的。令 $x \equiv r$ ， $y \equiv \frac{L}{D}z$ ， $u \equiv c$ ，则式 (7) ~ (10) 变为：

$$-\nabla \cdot x \nabla u = -x \phi^2 u \exp\left[\frac{\gamma\beta(1-u)}{1+\beta(1-u)}\right] \quad (11)$$

$$x=0, \quad \frac{\partial u}{\partial x} = 0 \quad (12)$$

$$x=1, \quad u=1 \quad (13)$$

$$y=0, \quad \frac{\partial u}{\partial z} = 0 \quad (14)$$

$$y=\frac{L}{D}, \quad u=1 \quad (15)$$

式 (11) ~ 式 (15) 是标准的非线性椭圆型 PDE 边值问题，可用函数 `pdenonlin()` 求解，其中主程序为 `CylindCat2.m`，几何 M 文件 `CylindCat2g.m` ($L/D=1$) 和边界条件 M 文件 `CylindCat2b.m` 同前所述由 `pdeTool` 得到。

当 $\beta=0$ (等温)，有 $f=-x\phi^2u$ ，此时 PDE 方程与上例相同，仅是求解域不同而已。

计算结果 在位置 $(r, z) = (0.394, 0.285)$ 、 $(0.394, 0.765)$ 、 $(0.803, 0.285)$ 、 $(0.803, 0.765)$ 处的浓度分别为：0.3350、0.5839、0.6479 和 0.7550。 $y = 0.75$ 时的 $c \sim x$ 关系图及 $c(x, y)$ 图形这里没有打印出（请读者运行该程序观察图形结果）。

5.6 小结

有限差分法 (FDM)

有限差分法是最基本的离散方法，虽然比较麻烦、且求解速度较慢，但它比较“万能”，一般离散问题都能有效解决。其中 Crank-Nicholson 隐式差分格式是 FDM 最常用的稳定算法。

正交配置法 (CM)

正交配置法对于几何对称的微分方程问题是一种比较有效的离散方法，其速度较快。

MOL 法

是求解一维动态和二维稳态 PDE 方程的简便方法，但必须注意步长要足够小，否则不收敛。对于一维动态方程（组），可直接用 MATLAB 函数 `pdepe()` 求解，它实际上是一种 MOL 法。对于二维稳态 PDE 方程（组），优先考虑 MATLAB PDE 工具箱的使用，但有的模型不能用 PDE 工具箱求解，这时，可考虑使用 MOL 法将偏微分方程组的边值问题转化为常微分方程组的初值问题。

有限元法 (FEM)

① 使用 `pdetool`——傻瓜式求解方式，无需编程，具有快速简便的优点，但 PDE 问题必须符合其标准形式，非标准形式的 PDE 问题，或者含有 3 个方程以上的 PDE 问题无法求解；

② 使用 MATLAB PDE 函数的编程方法——可解决含有 3 个方程以上的标准 PDE 问题。但对于非标准形式的 PDE 问题，使用这种方法比较困难。

几种方法 (FD、正交配置法、有限元法) 的比较

对于扩散-反应问题，正交配置法远远优于其他方法^[4]。如果问题不太困难，其解的梯度不陡，则使用正交配置法是空间近似的优先方法。若解的梯度较陡，则 FD 或 FEM 优先。

为了加快求解问题，选择方法的一般原则是：

对于一维动态模型，优先使用 MATLAB 函数 `pdepe()`。

对于二维稳态模型和二维动态模型：

若是含有 2 个方程以下的标准 PDE 问题，则优先选择 `pdetool` 方法；

若是含有 3 个方程以上的标准 PDE 问题，则优先选择 MATLAB PDE 函数的编程方法；

只有当 MATLAB PDE 工具箱不能解决时，才考虑使用正交配置法和有限差分法。

习 题

5-1 一维动态传质方程的求解^[6]

有一管状容器，高 20cm，开始容器内充有空气与乙醇的混合气，乙醇含量为 2%，在容器的底部是一个酒精池，乙醇蒸发不断挥发，在给定温度 30℃ 下乙醇液面处乙醇蒸气含量为 10%，在这管状容器上端，由于乙醇蒸气很快扩散到空气中，所以此处乙醇浓度基本为 0。如果仅考虑分子扩散，求乙醇浓度随时间 t 及位置 x 的变化。已知在 30℃ 下乙醇扩散系数为 $D = 0.119 \text{ cm}^2/\text{s}$ 。

数学模型

乙醇浓度随时间 t 及位置 x 的变化的 PDE 方程为

$$\frac{\partial C}{\partial t} = D \frac{\partial^2 C}{\partial^2 x}$$

$$\begin{aligned}\text{初始条件为} \quad & C(x, 0) = 2 \\ \text{边界条件为} \quad & C(0, t) = 0 \\ & C(20, t) = 10\end{aligned}$$

5-2 气体 A 通过液体 B 的扩散（一维动态扩散方程的求解）^[7]

(a) 如图 5-42 所示，在一容器中高为 $L = 10 \text{ cm}$ 的液体 B，在 $t = 0$ 时暴露于气体 A，其中组分 A 可在液体 B 中溶解，但与 B 不发生化学反应。在气液界面上 A 的浓度迅速达到 $C_{A0} = 0.01 \text{ mol/m}^3$ ，并保持恒定。已知 A 在 B 中的扩散系数为 $D_{AB} = 2 \times 10^{-9} \text{ m}^2/\text{s}$ 。请确定溶解在 B 中的 A 的浓度变化规律，并绘制溶解在 B 的 A 通量随时间的变化曲线。

(b) 重复 (a)，但考虑 A 与 B 之间发生化学反应： $A + B \rightarrow C$ ，反应速率为 $-r_A = 2 \times 10^{-7} C_A \text{ mol/s} \cdot \text{m}^3$ 。

数学模型

对 A 进行摩尔衡算，得到无反应时的模型 (1a) 和有反应时的模型 (1b)，这两种模型的初始条件和边界条件都是 (2) 和 (3) 式。

$$\text{无反应时,} \quad \frac{\partial C_A}{\partial t} = D_{AB} \frac{\partial^2 C_A}{\partial z^2} \quad (1a)$$

$$\text{有反应时,} \quad \frac{\partial C_A}{\partial t} = D_{AB} \frac{\partial^2 C_A}{\partial z^2} + k C_A \quad (1b)$$

$$\text{I.C.} \quad C_A(z, 0) = 0 \quad (2)$$

$$\text{B.C.} \quad \begin{cases} C_A(0, t) = C_{A0} \\ \left. \frac{\partial C_A}{\partial z} \right|_{z=L} = 0 \end{cases} \quad (3)$$

在 (b) 的情况，假设产品 C 的浓度可忽略不计，则扩散系数 D_{AB} 与 (a) 的情况一样。

A 进入液体 B 的摩尔扩散通量，可按 Fick 定律计算：

$$N_{Az}(t) = -D_{AB} \left. \frac{\partial C}{\partial z} \right|_{z=0} \quad (4)$$

5-3 孔中的扩散和吸附模型

在一理想孔中的扩散和吸附可由以下 PDE 方程组描述^[4]：

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2} - [k_a(1-f)c - k_d f], \quad 0 < x < 1, t > 0 \quad (1)$$

$$\frac{\partial f}{\partial t} = \beta [k_a(1-f)c - k_d f] \quad (2)$$

$$\text{I.C.} \quad c(x, 0) = f(x, 0) = 0 \quad 0 < x < 1 \quad (3)$$

$$\text{B.C.} \quad c(0, t) = 1, \quad t > 0 \quad (4)$$

$$\frac{\partial c}{\partial x}(1, t) = 0, \quad t > 0 \quad (5)$$

式中， C 为孔道内流体中被吸附剂的量纲一的浓度； f 为被吸附剂覆盖的孔分率； x 为量纲一的空间坐标； k_a 为吸附速率常数； k_d 为解吸速率常数， D, β 为常数。

已知 $D = \beta = k_a = 1$ ，分别计算 $k_a/k_d = 0.1$ 、1.0 和 10.0 时的浓度分布和温度分布，并作比较。

5-4 球形催化剂颗粒的质量平衡和热量平衡（传递）动态方程^[4]：

$$\frac{\partial y}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial y}{\partial r} \right) - \phi^2 R$$

$$Le \frac{\partial \theta}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial \theta}{\partial r} \right) + \beta \phi^2 R$$

$$\text{边界条件为:} \quad \text{在 } r=0 \text{ 处, } \frac{\partial y}{\partial r} = \frac{\partial \theta}{\partial r} = 0$$

$$\text{在 } r=1 \text{ 处, } y = \theta = 1$$

$$\text{初始条件为:} \quad t=0, \quad y=0, \quad \theta=1 \quad (0 \leq r \leq 1)$$

其中,

$$R = \exp\left[\gamma\left(1 - \frac{1}{\theta}\right)\right] y \quad (\text{一级})$$

已知 $\Phi = 1.0$, $\beta = 0.04$, $\gamma = 18$, 分别计算 $Le = 0.1$ 、 1.0 和 10.0 时催化剂颗粒中的浓度和温度分布, 并作比较。

5-5 二维稳态抛物型 PDE 边值问题^[8]

在管式反应器中在等温稳态条件下进行一级反应, 假设层流流动, 并忽略轴向扩散。试计算管式反应器的轴径向浓度分布。

数学模型

根据物料平衡推导得
$$-v_0 \left[1 - \left(\frac{r}{R} \right)^2 \right] \frac{\partial c}{\partial z} + D \left(\frac{\partial^2 c}{\partial r^2} + \frac{1}{r} \frac{\partial c}{\partial r} \right) - kc = 0 \quad (\text{a})$$

式中, v_0 为管中心流线的流速; R 为管径; k 为反应速度常数; c 为反应物浓度; D 为径向扩散系数; z 为距离管入口处的轴向位置; r 为距离管中心的位置。

引入中间变量:
$$Z = \frac{kz}{v_0}, \quad C = \frac{c}{c_0}, \quad \alpha = \frac{D}{kR^2}, \quad \xi = \frac{r}{R} \quad (\text{b})$$

式中, c_0 为进入反应器的反应物浓度。

则方程 (a) 变为无因次化方程:
$$(1 - \xi^2) \frac{\partial C}{\partial Z} = \alpha \left(\frac{\partial^2 C}{\partial \xi^2} + \frac{1}{\xi} \frac{\partial C}{\partial \xi} \right) - C \quad (\text{c})$$

边界条件:
$$Z = 0 \quad (\text{管入口处}), \quad C = 1.0 \quad (\text{d})$$

$$\xi = 0 \quad (\text{管中心}), \quad \frac{\partial C}{\partial \xi} = 0 \quad (\text{e})$$

$$\xi = 1.0 \quad (\text{管壁}), \quad \frac{\partial C}{\partial \xi} = 0 \quad (\text{f})$$

给定 $\alpha = 0.1$ 。

5-6 二维稳态 Dirichlet 型边界条件椭圆型 PDE 方程^[8]

有内热源 R 的二维稳态 PDE 方程 (Poisson 方程) 为:
$$k \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + R = 0 \quad (0 < x < 1.0, \quad 0 < y < 1.0),$$

式中 $k = 0.10$, $R = 4.0$ 。边界条件为: $x = 0, \quad T = 300^\circ\text{C},$

$$x = 1, \quad T = 300^\circ\text{C},$$

$$y = 0, \quad T = 100^\circ\text{C},$$

$$y = 1, \quad T = 100^\circ\text{C}.$$

试求解温度分布。

5-7 在有障碍的多孔介质中的流动^[9]

如图 5-43 所示, 在一个矩形区域 ABCD 的多孔介质中, 有一个渗透率为零的中心圆形区域 C 使流体不能渗透通过, 边界条件 AB 和 CD 也不能渗透流过。AB 和 CD 两端的压力分别为 $p = 10$ 和 $p = 0$ (任意单位)。假定 $c = k/\mu$ 为常数 1。

5-8 二维非稳态热传导^[10]

一个空心正方体燃烧室如图 5-44 所示, 燃烧室内侧尺寸为 $1\text{ m} \times 1\text{ m}$, 外侧尺寸为 $2\text{ m} \times 2\text{ m}$, 内侧温度保持在 700 K , 外墙壁保持在 300 K , 燃烧室墙壁的热扩散系数为 $\alpha = 5 \times 10^{-5} \text{ m}^2/\text{s}$, 导热系数为 $k = 1.2 \text{ W/m}$, 所有燃烧室的材料初始温度为 300 K 。

(1) 计算温度分布;

(2) 计算通过每米燃烧室内墙的热通量 q 。

已知数学模型:
$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

式中, T 为温度, K ; t 为时间, s ; $\alpha = \frac{k}{\rho c_p}$, k 为导热系数, W/m ; ρ 为密度, kg/m^3 ; c_p 为热容, J/kg 。

参 考 文 献

- 1 V. G. Jenson and G. V. Jeffeys, *Mathematical Methods in Chemical Engineering*, 2nd Edition, London: Academic Press Inc., 1977. 中译本, 台德荣译. 化工数学方法. 第二版. 北京: 化学工业出版社, 1982
- 2 Finlayson B A. *Nonlinear Analysis in Chemical Engineering*. New York: McGraw-Hill, 1980
- 3 张建侯, 许锡恩编著. 化工过程分析与计算机模拟. 北京: 化学工业出版社, 1989, 227~246, 287~289
- 4 Mark E. Davis, *Numerical Methods and Modeling for Chemical Engineers*, John Wiley & Sons, Inc., 1984. 215, 219~221
- 5 徐自新. 微分方程近似解. 上海: 华东化工学院出版社, 1990, 141
- 6 日本化学工学协会编, 麻德贤译. 化学工程程序设计例题与习题集. 北京: 化学工业出版社, 1984
- 7 Alkis Constantinides and Navid Mostoufi, *Numerical Methods for Chemical Engineers with MATLAB Application*, Prentice Hall, Upper Saddle River, NJ, 1999
- 8 Leon Lapidus, *Digital computer for chemical engineers*, McGraw-Hill, 1962, 177 例 4.1, 178 例 4.2
- 9 James O. Wilkes, *Fluid Mechanics for Chemical Engineers*. Prentice Hall PTR, 1999
- 10 M. B. Cutlip and M. Shacham, *Problem Solving in Chemical Engineering*, 1999, 254
- 11 张吉瑞编著. 化工数学方法. 北京: 中国石化出版社, 1995
- 12 *Partial Differential Equation Toolbox User's Guide*, Version 1.04, The MathWorks, Inc., 2002 (pde.pdf)
- 13 李涛, 贺勇军, 刘志敏等编著. *Matlab 工具箱应用指南—应用数学篇*. 北京: 电子工业出版社, 2000

第6章 化工最优化方法

随着科学技术,尤其是计算机技术的发展,最优化方法已经在各个领域,如化学工程、生化工程、机械工程、土木工程、经济管理等,得到越来越广泛的应用。最优化方法在化学工程中的应用,主要涉及研究与开发中的实验方案最优化、化工数学模型的参数估计或辨识、化工过程优化设计、工艺操作参数的优化、过程优化控制以及最优生产调度等等。

本章将介绍最优化方法及其常用算法,并在6.2和6.3中分别叙述化工过程设计优化和操作优化的应用实例。有关实验方案的最优化将在第8章中介绍,而最优化方法在数学模型参数估计方面的应用在第7章中介绍。

6.1 最优化方法及其常用算法

6.1.1 最优化方法概述

6.1.1.1 最优化问题的基本概念

目标函数 凡是最优化问题,都涉及最优目标,如使产品收率最大、使能耗最小等等。把目标写成数学形式的表达式称为目标函数。

约束条件和状态方程 变量的取值范围通常都有一个限制,这种对变量取值的限制称为约束条件。凡约束条件是以不等式进行表达的称为不等式约束,用等式来表达的称为等式约束。在化工过程最优化问题中,物料衡算式、热量衡算式、动量衡算式等都属于等式约束,或称为过程或系统的状态方程。

决策变量和状态变量、系统自由度 状态变量是能够描述过程或系统的特征或行为的(即状态的)一组变量,其值是不能任意选取的。**决策变量**根据最优化任务的不同又称为设计变量、控制变量、操作变量,它是由决策者(设计者、控制者、操作者等)根据目标和约束条件的要求而确定的。所谓决策,就是选择一组决策变量以满足任务的要求。决策变量已经确定,状态变量也就随之确定,从而过程或系统的状态也就被确定。在化工过程中,通常把能控制的变量作为决策变量,如压力、温度、流量等易于测量和控制,可作为决策变量。在最优化问题中,决策变量数又称为系统的自由度。在确定状态变量和决策变量时必须遵循的原则:状态变量数 = 状态方程数,变量的总数 - 状态方程数 = 决策变量数。

6.1.1.2 最优化问题的一般形式

各种各样的优化问题具有非常相似的结构,最优化问题的一般形式为:

$$\min_{x \in R^n} J = f(x) \quad (6-1a)$$

$$\text{s.t. } G(x_i) = 0, \quad i = 1, 2, \dots, m_e \quad (6-1b)$$

$$G(x_i) \leq 0, \quad i = m_e + 1, \dots, m \quad (6-1c)$$

$$x_l \leq x \leq x_u \quad (6-1d)$$

其中, \mathbf{x} 为设计变量向量: $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, 方程(6-1a)为目标函数(利润函数、成本函数等), (6-1b)为等式约束(方程), (6-1c)为不等式约束(方程), (6-1d)为参数边界。

一个优化解就是满足(6-1b)~(6-1d)的情况下,使函数(6-1a)达到最优的解。

以 m_e 表示独立等式约束方程的个数, 以 m_i 表示独立不等式约束方程的个数, 若未知变量数等于 $m_e + m_i$, 则不管优化准则如何, 至少存在一个解。当 (6-1b) 和 (6-1c) 中的模型由非线性关系组成时, 存在多个解。如果只有惟一解, 就没必要使用最优化求解, 而仅仅解方程组即可。

根据变量、目标函数和约束条件的不同, 最优化问题可分为:

- ① 线性优化; ② 二次优化; ③ 非线性优化; ④ 多目标优化。

6.1.1.3 优化问题一般求解方法^[1]

没有哪种优化算法对所有的优化问题都有效, 对于某一特定的优化问题, 应根据原问题的目标函数和约束条件的特点以及自变量和因变量的数目, 判断原问题属于哪一类型的优化问题 (线性优化、二次优化或非线性优化等), 再选择相应的优化算法进行求解。

求解优化问题的六个步骤:

- ① 分析过程以确定过程变量及感兴趣的具体特征, 并列出所有的变量表;
- ② 确定优化准则, 根据①的变量表及有关系数给定目标函数 (性能模型);
- ③ 用数学表达式写出过程模型或设备模型, 使过程输入-输出变量与有关系数相关联。使用已知的物理原理 (质量平衡、热量平衡), 经验关联式、隐含概念, 和外部约束等等。识别自变量和因变量以获得自由度的数目;

④ 如果问题形式范围太大, 则 (a) 把问题分成可管理的部分; (b) 简化目标函数和模型;

- ⑤ 选用合适的优化技术进行求解;

- ⑥ 检查答案, 并检查结果对问题中系数和假设变化的灵敏度。

6.1.1.4 MATLAB 最优化工具箱的最小化函数

MATLAB 最优化工具箱提供的最小化函数见表 6-1。

表 6-1 最小化函数

函 数	描 述	函 数	描 述
fgoalattain	多目标达到问题	fminsearch, fminunc	无约束非线性最小化
fminbnd	有边界的单变量非线性最小化	fseminf	半无限问题
fmincon	有约束的非线性最小化	linprog	线性规划
fminimax	最大最小化	quadprog	二次规划

用 MATLAB 最优化工具箱求解的一般步骤是: 首先建立最优化的数学模型 (包括优化目标函数和约束条件), 然后选择适合的最优化方法及相应的 MATLAB 最优化函数进行求解。下面具体地介绍常见的最优化方法、相应的 MATLAB 最优化函数及其应用。

6.1.2 单变量最优化问题

数学模型:

$$\min_x f(x) \quad (6-2a)$$

$$x_1 < x < x_2 \quad (6-2b)$$

式中, x_1 、 x_2 和 x 为标量, $f(x)$ 为目标函数。

单变量 (一元) 函数的最优化方法又称一维搜索法, 根据目标函数是否需要求导, 可分为直接法和间接法两种, 前者不需要对目标函数求导, 收敛速度较慢, 而后者则需要用到目标函数的导数, 但收敛速度较快。直接法主要有黄金分割法、二次多项式近似法和三次多项式近似法等。间接法主要包括牛顿切线法、割线法和三次插值法等。其中, 黄金分割法算法

简单而稳定，三次插值法收敛速度很快，若目标函数容易求导，可优先考虑使用三次插值法。

MATLAB 优化工具箱提供的最优化方法有：黄金分割法、二次插值法、三次插值法、二次、三次混合插值法等。

MATLAB 常用的单变量最优化函数为 `fminbnd()`，其算法基于黄金分割法和二次插值法。

函数 `fminbnd()` 的用法

调用格式：`x = fminbnd(@fun, x1, x2)` % 搜索函数 `fun` 在区间 `[x1, x2]` 内的最小值
`[x, fval, exitflag] = fminbnd(@fun, x1, x2, options, p1, p2, ...)`

输入参数：`fun` 目标函数的函数名，`fun` 定义为：

`function f = fun(x, p1, p2, ...)`
`f = ...`

`x1, x2` 区间 `[x1, x2]`

`options` 优化参数选项

`p1, p2, ...` 需要传递的参数

输出参数：`x` 最优解

`fval` 目标函数的最小值

`exitflag` 描述退出条件：`exitflag > 0` 表示目标函数收敛于解 `x` 处；
`exitflag = 0` 表示已经达到函数评价或迭代的最大次数；
`exitflag < 0` 表示目标函数不收敛。

【例 6-1】 在区间 `[0 100]` 内求取单变量函数的最小值^[3]

$$\min f(x) = x^3 + 3x^2 - 9x$$

程序说明 这是单变量最优化问题，程序用 `fminbnd()` 求解，其中 `ObjFunc()` 定义目标函数。

程序清单 `xFminbnd.m`

```
function xFminbnd                                % 单变量最优化--函数 fminbnd() 的简单应用示例
clear all; clc
x1 = 0; x2 = 100;
[x, fval] = fminbnd(@ObjFunc, x1, x2);
fprintf(' \nResults:\n'), fprintf('   Optimum solution: %f\n', x)
fprintf('   Objective value: %f', fval)
% -----
function f = ObjFunc(x)
f = x^3 + 3*x^2 - 9*x;
```

计算结果 最优解：`x = 1`，目标值：`f(x) = -5`。

6.1.3 线性规划

线性规划问题即带线性约束的线性目标函数最小化问题。线性规划的标准形式是

$$\min_{\mathbf{x}} \mathbf{f}^T \mathbf{x} \quad (6-3a)$$

$$s.t. \quad \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \quad (6-3b)$$

$$\mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq} \quad (6-3c)$$

$$\mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub} \quad (6-3d)$$

式中, f 、 x 、 b 、 beq 、 lb 和 ub 为向量; A 和 Aeq 为矩阵。

线性规划的算法有单纯形法、两步法、改进单纯形法和对偶单纯形法等。

MATLAB 优化工具箱提供的函数 `linprog()`, 用于求解线性规划问题 (6-3), 其算法采用投影法 (projection method), 它是改进单纯形法的一种算法。对大型优化问题, `linprog()` 采用 LIPSOL 法 (Linear Interior Point Solver), 它是 Mehrotra 预测-校正法的一种变化形式。对中型优化问题, `linprog()` 使用的是投影法, 即函数 `quadprog()` 中所用的算法。

函数 `linprog()` 的用法

调用格式: $x = \text{linprog}(@f, A, b)$

$x = \text{linprog}(@f, A, b, Aeq, beq, lb, ub, x0, options)$

$[x, fval, exitflag, output, lambda] = \text{linprog}(@f, A, b)$

其中输入参数表中的变量与式 (6-3a) ~ (6-3d) 中的变量一致。输出参数表中的变量 x , $fval$, $exitflag$, $output$, $lambda$ 等与前面相同。

【例 6-2】 求解下列线性规划问题^[3]

$$\begin{aligned} \max \quad & -5x_1 + 6x_2 + 7x_3 \\ \text{s.t.} \quad & 5x_1 - 6x_2 + 10x_3 \leq 20 \\ & x_1 + x_2 + x_3 = 5 \\ & x_1 + 5x_2 - 3x_3 \geq 15 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

程序说明 原问题不是标准形式, 要先把原问题转化为形如式 (6-3) 的最小化标准形式:

$$\begin{aligned} \min \quad & 5x_1 - 6x_2 - 7x_3 \\ \text{s.t.} \quad & 5x_1 - 6x_2 + 10x_3 \leq 20 \\ & -x_1 - 5x_2 + 3x_3 \leq -15 \\ & x_1 + x_2 + x_3 = 5 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

然后才调用函数 `linprog()` 求解。

程序清单 `x1LinProg.m`

```
% 线性规划--函数 linprog() 的简单应用示例
clear all; clc
f = [5 -6 -7]';
A = [5 -6 10
     -1 -5 3
     0 0 0];
b = [20 -15 0]'; Aeq = [1 1 1]; beq = [5]; lb = zeros(3, 1);
[x, fval, exitflag, output, lambda] = linprog(f, A, b, Aeq, beq, lb)
```

计算结果 最优解: $x = [0.0000 \ 3.7500 \ 1.2500]'$, 函数最小值: -31.2500。

【例 6-3】 求解线性规划问题^[3]

$$\min \quad -2x_1 - x_2 + x_3$$

$$\begin{aligned} \text{s.t. } & x_1 + x_2 + 2x_3 = 6 \\ & x_1 + 4x_2 - x_3 \leq 4 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \leq 5 \end{aligned}$$

程序说明 原问题不是标准形式，要先把原问题转化为最小化标准形式：

$$\begin{aligned} & \min f(x) \\ \text{s.t. } & AX \leq b \\ & A_{eq}X = b_{eq} \\ & LB \leq X \leq UB \end{aligned}$$

即以 $(-x_3)$ 代替 x_3 ，最优化模型变为标准形式：

$$\begin{aligned} & \min -2x_1 - x_2 - x_3 \\ \text{s.t. } & x_1 + 4x_2 + x_3 \leq 4 \\ & x_1 + x_2 - 2x_3 = 6 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq -5 \end{aligned}$$

此问题可用 *LinProg* 函数求解，则原问题 x_3 的最优解应等于 $(-x_3)$ ，其他不变。

程序清单 见光盘中的 *x2LinProg.m*。

计算结果 最优解： $x = [4.6667 \quad 0.0000 \quad 0.6667]'$ ，函数最小值： -8.6667 。

6.1.4 无约束多变量问题最优化

无约束多变量最优化问题即无约束非线性规划问题，其一般形式为

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (6-4)$$

式中， \mathbf{x} 为一向量， $f(\mathbf{x})$ 为一函数。

求解无约束多变量最优化问题的方法有两大类，即直接搜索法（Line Search method）和梯度法（Gradient method）。

直接搜索法适用于目标函数高度非线性、没有导数或导数不易求出的情况。该方法在迭代过程中只用到目标函数值，而无需计算一阶导数和二阶导数，因此具有很强的适应性，但收敛速度慢。直接搜索法包括单纯形法、鲍威尔（Powell）法、Hook-Jeeves 搜索法和 Pavell 共轭梯度方向法等。

梯度法收敛速度很快，但要求计算目标函数的梯度（一阶导数）和 Hessian 矩阵（二阶导数）。常见的梯度法有最速下降法（Steepest Descent Method）、共轭梯度法、（阻尼）牛顿法、拟牛顿法（Quasi-Newton Methods）和 Marquart 法等。这几种算法的特点如下^[5]。

（1）最速下降法-在远离极值点时非常有效，但当接近极值点时，收敛速度非常缓慢。

（2）共轭梯度法-是最速下降法的一种改进，其收敛速度较快，具有超线性收敛或更高一些，而且计算简单、存储量小，适用于维数较高的情况。但其数值稳定性有时不如变尺度法。

（3）牛顿法和阻尼牛顿法-牛顿法的收敛速度很快，但对初始点的选择比较苛刻。阻尼牛顿法对牛顿法作了改进，既保持了牛顿法收敛速度快的优点，又对初始点没有过于苛刻的要求。牛顿法和阻尼牛顿法的缺点是：每次迭代需要计算目标函数的 Hessian 矩阵及其逆矩阵，因而计算量和存储量都比较大。此外，要求 Hessian 矩阵必须是非奇异的，否则无法继续迭代。实际上，即使是奇异的，也不能保证算法一定收敛。

（4）拟牛顿法（Quasi-Newton method）它既保留了牛顿法收敛速度快的优点，又通过构造一个 Hessian 矩阵的近似矩阵免去了目标函数的 Hessian 矩阵及其逆矩阵的计算。因此，

它成为求解无约束多变量最优化问题最有效的方法之一。拟牛顿法取决于 Hessian 及其逆阵的近似公式，有 BFGS (Broyden-Fletcher-Goldfarb-Shanno) 和 DFP (Davidon-Fletcher-Powell) 两种算法，它们都具有超线性收敛。对于一般的问题，BFGS 方法是构造一个 Hessian 矩阵的近似矩阵最有效的方法，它比 DFP 更具优越。

(5) 马夸特 (Marquart) 法—适用于平方和形式的函数最小化问题 (即最小二乘问题)。平方和形式的目标函数为：

$$\min_{\mathbf{x}} \sum_i f_i(\mathbf{x})^2 + C \quad (6-5)$$

原则上这类问题可以用前面几种方法求解，但由于目标函数具有平方和形式，用高斯-牛顿法 (Gauss-Newton Method) 和 Levenberg-Marquart 法，更简单有效。此最小二乘问题将在 6.1.8 中专门介绍。

MATLAB 优化工具箱提供无约束多变量优化问题 (6-4) 的求解函数有：fminunc() 和 fminsearch()，其算法分别采用拟牛顿法 (Quasi-Newton method) 和 Nelder-Mead 单纯形法 (直接搜索法)。对于平方和形式的目标函数最小化问题 (6-5)，MATLAB 优化工具箱提供的求解函数为 lsqnonlin()，将在 6.1.8 中叙述。下面介绍函数 fminunc() 和 fminsearch() 的用法。

函数 fminunc() 的使用方法

调用格式：
 $\mathbf{x} = \text{fminunc}(@\text{fun}, \mathbf{x}_0)$
 $\mathbf{x} = \text{fminunc}(@\text{fun}, \mathbf{x}_0, \text{options}, p_1, p_2, \dots)$
 $[\mathbf{x}, \text{fval}, \text{exitflag}] = (\text{fminunc}, @\text{fun}, \mathbf{x}_0, \text{options}, p_1, p_2, \dots)$

输入参数：
 fun 目标函数的函数名
 \mathbf{x}_0 初值，可以是标量、向量或矩阵
 options 优化参数选项

输出参数：
 \mathbf{x} 最优解
 fval 目标函数的最小值
 exitflag 同 fminbnd()

函数 fminsearch() 的使用方法

使用方法与函数 fminunc() 完全相同。

【例 6-4】 用 Nelder-Mead 单纯形法求 Rosenbrock 的“香蕉函数” $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ 的极小值。初值取 $[-1.9, 2]$ 。

程序说明：先定义目标函数 ObjFunc()，然后调用 fminsearch() 即可，见 xFminsearch.m。

6.1.5 二次规划

若某非线性规划的目标函数是关于自变量的二次函数，且约束条件是线性函数，则这种规划称为二次规划。其数学模型为

$$\min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{f}^T \mathbf{x} \quad (6-6a)$$

$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \quad (6-6b)$$

$$\mathbf{A}_{\text{eq}} \cdot \mathbf{x} = \mathbf{b}_{\text{eq}} \quad (6-6c)$$

$$\mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub} \quad (6-6d)$$

式中， \mathbf{H} , \mathbf{A} 和 \mathbf{A}_{eq} 为矩阵， \mathbf{x} , \mathbf{f} , \mathbf{b} , \mathbf{b}_{eq} , \mathbf{lb} 和 \mathbf{ub} 为向量。

二次规划问题 (6-6) 的 MATLAB 求解函数为 quadprog()。

函数 quadprog()

调用格式: $x = \text{quadprog}(H, f, A, b, Aeq, beq)$

$x = \text{quadprog}(H, f, A, b, Aeq, beq, lb, ub, x0)$

$[x, fval, exitflag] = \text{quadprog}(H, f, A, b, Aeq, beq, lb, ub, x0, options, p1, p2, \dots)$

$[x, fval, exitflag, output, lambda] = \text{quadprog}(\dots)$

输入参数: $H, f, A, b, Aeq, beq, lb, ub$ 与模型方程 (6-6) 中的符号一致。 $x0$ 为初始值,

输出参数: $x, fval, exitflag, output$ 同函数 $fminbnd()$ 。

【例 6-5】 求解二次规划问题^[3]: $\min \frac{1}{2}(5x_1^2 + 6x_1x_2 + 5x_2^2) - 95x_1 - 105x_2$

$$s. t. \quad x_1 + 2x_2 \leq 10$$

$$3x_1 + x_2 \leq 15$$

$$2x_1 + 3x_2 \leq 30$$

$$-15x_1 + 13x_2 \leq 0$$

$$x_1 \geq 0, \quad x_2 \geq 0$$

程序说明 原问题不是标准形式, 要先把原问题转化为形如式 (6-6) 的二次规划标准形式, 则可得到有关参数: H, f, A, b, lb, ub , 其中 $x = [x_1; x_2]$ 。

程序清单 *x1QuadProg.m*

```
clear all; clc % 二次规划 (优化) 问题的求解例子
H = [5 3; 3 5]; f = [-95; -105]; A = [1 2; 3 1; 2 3; -15 13]; b = [10; 15; 30; 0]; lb = [0; 0];
[x, fval, exitflag, output, lambda] = quadprog(H, f, A, b, [], [], lb)
```

计算结果 最优解为 $x = [4 \ 3]$, 目标函数最小值为 $fval = -596.5000$ 。

【例 6-6】 求解二次规划问题

$$\min 2.5x_1^2 - 35x_1 - 56x_2$$

$$s. t. \quad -3x_1 + x_2 = 7$$

$$8x_1 + 3x_2 \leq 12$$

$$-3x_1 + 4x_2 \geq -4$$

程序说明 原问题不是标准形式, 要先把原问题转化为形如式 (6-6) 的二次规划标准形式 (原问题第三个约束条件两边乘以 -1)。

程序清单 见光盘中的 *x2QuadProg.m*。

计算结果 最优解为 $x = [0.52941 \ 5.41176]$, 目标值为 $fval = -211.31142$ 。

【例 6-7】 求解最优化问题^[3]

$$\min 3x_1^2 - 1.5x_2^2 - 5x_1 + 7x_2$$

$$s. t. \quad 3x_1 - 6x_2 \leq 35$$

$$9x_1 + 5x_2 \leq 56$$

$$x_1 \geq 2$$

$$x_2 \geq 0$$

程序说明 此问题没有等式约束条件, 调用 $\text{quadprog}()$ 时, $Aeq = [], beq = []$ 。

程序清单 见光盘中的 *x3QuadProg.m*。

计算结果 最优解: $x = [2 \ 0]$, 目标值: $fval = 2$ 。

6.1.6 多变量有约束最优化（非线性规划）问题

多变量有约束非线性最优化问题也称有约束非线性规划问题，它是最困难的最优化问题之一。求解此类优化问题的算法有以下几种。

(1) 序贯二次规划法 (Sequential Quadratic Programming Method, 简称 SQP 法) — 它是无约束极值问题的拟牛顿法在有约束极值问题中的深化和推广，是当今求解光滑的非线性规划问题的最有效的算法之一。

(2) 可行方向法约旦狄克 (Zoutendijk) 可行方向法，托-文 (Topkis-Veinott) 法和若森 (Rosen) 投影梯度法。

(3) 复合形法—是单纯形法对约束问题的推广。MATLAB 优化工具箱提供的非线性规划求解函数为 `fmincon()`，它采用最有效的序贯二次规划法 (SQP)。

函数 `fmincon()`

规划问题：

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (6-7a)$$

$$s.t. \quad \mathbf{c}(\mathbf{x}) \leq 0 \quad (6-7b)$$

$$\mathbf{ceq}(\mathbf{x}) = 0 \quad (6-7c)$$

$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \quad (6-7d)$$

$$\mathbf{Aeq} \cdot \mathbf{x} = \mathbf{beq} \quad (6-7e)$$

$$\mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub} \quad (6-7f)$$

式中， \mathbf{x} , \mathbf{b} , \mathbf{beq} , \mathbf{lb} 和 \mathbf{ub} 为向量， \mathbf{A} 和 \mathbf{Aeq} 为矩阵， $\mathbf{c}(\mathbf{x})$ 和 $\mathbf{ceq}(\mathbf{x})$ 为返回向量的函数。 $f(\mathbf{x})$ 为返回标量的函数， $f(\mathbf{x})$, $\mathbf{c}(\mathbf{x})$ 和 $\mathbf{ceq}(\mathbf{x})$ 可以是非线性函数。

调用格式：`x = fmincon(@fun, x0, A, b)`

`x = fmincon(@fun, x0, A, b, Aeq, beq, lb, ub, @nonlcon, options, p1, p2, ...)`

`[x, fval, exitflag, output, lambda, grad, hessian] = fmincon(@fun, x0, ...)`

其中，输入变量 \mathbf{A} , \mathbf{b} , \mathbf{Aeq} , \mathbf{beq} , \mathbf{lb} , \mathbf{ub} 等与式 (6-7d) ~ (6-7f) 中的变量一致。函数 `nonlcon` 用于定义非线性不等式约束 (6-7b) 和非线性等式约束 (6-7c)，它要求输入一个向量 \mathbf{x} ，返回两个变量—解 \mathbf{x} 处的非线性不等式向量 \mathbf{c} 和非线性等式向量 \mathbf{ceq} 。例如：

```
function [c, ceq] = nonlcon(x)
```

```
c = ... % 计算解 x 处的非线性不等式;
```

```
ceq = ... % 计算解 x 处的非线性等式。
```

若还计算约束的梯度，需用下列命令设置 'GradConstr' 选项：

```
options = optimset('GradConstr', 'on')
```

这时，函数 `nonlcon` 还必须在第三个和第四个输出变量中返回 $\mathbf{c}(\mathbf{x})$ 的梯度 \mathbf{GC} 和 $\mathbf{ceq}(\mathbf{x})$ 的梯度 \mathbf{GCeq} ，例如

```
function [c, ceq, GC, GCeq] = nonlcon(x)
```

```
c = ... % 计算解 x 处的非线性不等式;
```

```
ceq = ... % 计算解 x 处的非线性等式。
```

```
If nargout > 2 % 当要求大于 2 个的输出变量（参数）时
```

```
GC = ... % 不等式的梯度
```

```
GCeq = ... % 等式的梯度
```

```
end
```

【例 6-8】 求解最优化问题^[3]

$$\begin{aligned} \min \quad & -x_1 + 10x_1^2 + 10x_2^2 \\ \text{s.t.} \quad & x_1^2 + x_2^2 - 1 = 0 \end{aligned}$$

初值为 $x_0 = [0.8; 0.6]$ 。

程序说明 函数 `func()` 定义目标函数, `NlinCons()` 定义等式非线性约束方程。

程序清单 `x1Fmincon.m`

```
function x1Fmincon
clear all; clc
x0 = [0.8; 0.6];
[x, fval] = fmincon(@func, x0, [], [], [], [], [], [], @NlinCons)
% -----
function f = func(x)
f = -x(1) + 10*x(1)^2 + 10*x(2)^2;
% -----
function [c, ceq] = NlinCons(x)
ceq = x(1)^2 + x(2)^2 - 1;
```

计算结果 最优解: $x = [1 \ 0]$, 目标值: $fval = 9.0000$ 。

【例 6-9】 求解最优化问题^[3]

$$\begin{aligned} \min \quad & -x_1 - x_2 \\ \text{s.t.} \quad & x_1^2 - x_2 \leq 0 \\ & x_1^2 + x_2^2 - 1 = 0 \end{aligned}$$

初值为 $x_0 = [0.5 \ 1]$ 。

程序说明 函数 `ObjFunc()` 定义目标函数, `NlinCons()` 定义不等式和等式非线性约束条件。

程序清单 见光盘中的 `x2Fmincon.m`。

计算结果 最优解: $x = [0.70711 \ 0.70711]$, 目标值: $fval = -1.41421$ 。

6.1.7 多目标最优化

前面介绍的最优化方法只有一个目标函数, 是单目标最优化方法。但是, 有时实际工程问题希望多个指标都达到最优值, 即它有多目标函数, 这就是多目标最优化问题, 亦即多目标规划 (Vector Programming, 简称 VP) 问题, 其数学模型为

$$\min_{x \in R^n} F(x) \quad (6-8a)$$

$$G_i(x) = 0 \quad i = 1, \dots, m_e \quad (6-8b)$$

$$G_i(x) \leq 0 \quad i = m_e + 1, \dots, m \quad (6-8c)$$

$$x_l \leq x \leq x_u \quad (6-8d)$$

式中, $F(x)$ 为目标函数向量。

多目标问题最优化的 MATLAB 函数为 `fgoalattain()`。

函数 fgoalattain()

数学模型:

$$\begin{aligned}
& \min_{\mathbf{x}, \gamma} \\
& F(\mathbf{x}) - \text{weight} \cdot \gamma \leq \text{goal} \\
& c(\mathbf{x}) \leq 0 \\
& \text{ceq}(\mathbf{x}) = 0 \\
& \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \\
& \mathbf{Aeq} \cdot \mathbf{x} = \mathbf{beq} \\
& \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub}
\end{aligned} \tag{6-9}$$

式中， \mathbf{x} 、 weight 、 goal 、 \mathbf{b} 、 \mathbf{beq} 、 \mathbf{lb} 和 \mathbf{ub} 为向量； \mathbf{A} 和 \mathbf{Aeq} 为矩阵； $c(\mathbf{x})$ 、 $c_{eq}(\mathbf{x})$ 和 $F(\mathbf{x})$ 为函数；返回向量； $F(\mathbf{x})$ 、 $c(\mathbf{x})$ 和 $c_{eq}(\mathbf{x})$ 可以为非线性函数。

调用格式：

```

x = fgoalattain(@fun, x0, goal, weight)
x = fgoalattain(@fun, x0, goal, weight, A, b, Aeq, beq, lb, ub, @nonlcon, options, p1, p2, ...)
[x, fval, attainfactor, exitflag, output, lambda] = fgoalattain(@fun, x0, ...)

```

输入参数：

fun 目标函数
x0 初值
goal 指定目标，即目标希望达到的向量值。该向量的长度与 fun 函数返回的目标数 F 相等。 $\text{fgoalattain}()$ 试图通过最小化向量 F 中的值来达到 goal 参数给定的目标。
weight 权重参数
A, b, Aeq, beq, lb, ub 与标准形式的方程 (6-8) 相同。
nonlcon 定义非线性约束条件的函数名

输出参数：

x 最优解
attainfactor 解 \mathbf{x} 处的目标达到因子，即变量是超过或低于目标的个数。若 attainfactor 为负，则目标已经溢出；若 attainfactor 为正，则目标个数还未达到。

【例 6-10】多目标最优化^[4]

某化工厂拟生产两种新产品 A 和 B，其生产设备费用分别为：产品 A 2 万元/吨；产品 B 5 万元/吨。这两种产品均将造成环境污染，设由公害所造成的损失可折算为：产品 A 4 万元/吨；产品 B 1 万元/吨。由于条件限制，工厂生产产品 A 和 B 的最大生产能力各为每月 5 吨和 6 吨，而市场需要这两种产品的总量每月不少于 7 吨。试问工厂如何安排生产计划，在满足市场需要的前提下，使设备投资和公害损失均达最小。该工厂决策认为，这两种目标中环境污染应优先考虑，设备投资的目标值为 20 万元，公害损失的目标为 12 万元。

数学模型 设工厂每月生产产品 A 为 x_1 吨，B 为 x_2 吨，设备投资费为 $f_1(x)$ ，公害损失费为 $f_2(x)$ ，则这个问题可表达为多目标优化问题：

$$\begin{cases}
\min f_1(x) = 2x_1 + 5x_2 \\
\min f_2(x) = 4x_1 + x_2 \\
x_1 \leq 5 \\
x_2 \leq 6 \\
x_1 + x_2 \geq 7 \\
x_1, x_2 \geq 0
\end{cases}$$

程序说明 函数 ObjFun()定义目标函数向量。已知目标为: goal = [20, 12], 权重按目标比例确定: weight = [20, 12], 初值为 x0 = [2, 5]。将此问题的约束条件与 (6-9) 式标准约束条件比较, 可得到约束条件的系数 A、b 和 lb。这样, 调用 fgoalattain()即可求解。

程序清单 xFgoalattain.m

```
function xFgoalattain
clear all; clc
goal = [20, 12]; weight = [20, 12]; x0 = [2, 5]; % 给定目标, 权重按目标比例确定, 给出初值
A = [1, 0; 0, 1; -1, -1]; b = [5, 6, -7]; lb = zeros(2,1); % 给出约束条件的系数
[x,fval,attainfactor,exitflag] = fgoalattain(@ObjFun,x0,goal,weight,A,b,[],lb,[]) % 求解
% -----
function f = ObjFun(x)
f(1) = 2*x(1)+5*x(2); f(2) = 4*x(1)+x(2);
```

计算结果 x = [2.917 4.083], 即工厂每月应生产 2.917 吨的产品 A 和 4.083 吨的产品 B, 相应的设备投资费和公害损失费分别为 26.25 万元和 15.75 万元。

6.1.8 最小二乘法

6.1.8.1 最小二乘问题简介

最小二乘问题的数学模型

在 6.1.4 中提及过最小二乘问题, 即式 (6-5), 它可写成如下向量形式

$$\min_x f(x) = \frac{1}{2} \|F(x)\|_2^2 = \frac{1}{2} \sum_i F_i(x)^2 \quad (6-10)$$

式中, x 为一向量; F(x) 为返回向量值的函数, 即 $F(x) = [f_1(x), f_2(x), \dots]$, $f_i(x)$ 为函数。

这类问题在数据拟合、非线性参数估计等科学与工程计算中很常见, 其解法有 Gauss-Newton (简称 G-N) 法和 Levenberg-Marquart (简称 L-M) 法。L-M 法又称阻尼最小二乘法, 它是在 G-N 法的基础上加入阻尼因子的一种改进方法。当阻尼因子为 0 时, L-M 法即为 G-N 法, 当阻尼因子趋于 ∞ 时, 即为最速下降法。G-N 法的收敛速度较快, 但有时会出现问题, L-M 法可解决此问题, 比较稳定, 其效率低于 G-N 法, 而大大高于最速下降法。

MATLAB 最小二乘法函数如表 6-2 所示。

表 6-2 最小二乘法函数表

函 数		描 述
线性	\	线性最小二乘
	lsqlin	有约束线性最小二乘
	lsqnonneg	非负线性最小二乘
非线性	lsqcurvefit	非线性曲线拟合
	lsqnonlin	非线性最小二乘

6.1.8.2 线性最小二乘问题

线性最小二乘问题 (Linear Least Squares of matrix problems): \

与线性方程求解一样, 用左除 (“\”) 算符可以求解线性最小二乘问题:

$$X = A \setminus b \quad (6-11)$$

非负线性最小二乘问题 (Nonnegative least squares problem): lsqnonneg()

非负线性最小二乘问题的数学模型为

$$\min_x f(x) = \frac{1}{2} \|Cx - d\|_2^2 \quad (6-12a)$$

$$x \geq 0 \quad (6-12b)$$

式中, 矩阵 C 和向量 d 为目标函数的系数。独立变量向量 x 要求非负。

求解此问题的对应函数为 lsqnonneg()。

函数 lsqnonneg()

调用格式: $x = \text{lsqnonneg}(C, d)$

$x = \text{lsqnonneg}(C, d, x0, \text{options})$

$[x, \text{resnorm}, \text{residual}, \text{exitflag}, \text{output}, \text{lambda}] = \text{lsqnonneg}(\dots)$

其中, 变量 C, d, x 同式 (6-12a), 其他变量同前。

有约束线性最小二乘问题 (Constrained linear least squares problem): lsqlin()

有约束线性最小二乘问题的数学模型为

$$\min_x f(x) = \frac{1}{2} \|Cx - d\|_2^2 \quad (6-13a)$$

$$\text{s.t. } A \cdot x \leq b \quad (6-13b)$$

$$A_{\text{eq}} \cdot x = \text{beq} \quad (6-13c)$$

$$lb \leq x \leq ub \quad (6-13d)$$

式中, C, A 和 A_{eq} 为矩阵; d, b, beq, lb, ub 和 x 为向量。

求解此问题的对应函数为 lsqlin()。

函数 lsqlin()

调用格式: $x = \text{lsqlin}(C, d, A, b)$

$x = \text{lsqlin}(C, d, A, b, A_{\text{eq}}, \text{beq}, lb, ub, x0, \text{options}, p1, p2, \dots)$

$[x, \text{resnorm}, \text{residual}, \text{exitflag}, \text{output}, \text{lambda}] = \text{lsqlin}(C, d, A, b)$

6.1.8.3 非线性最小二乘问题

非线性最小二乘问题 (nonlinear least squares problem) 的 MATLAB 求解函数主要包括非线性最小二乘法函数 lsqnonlin() 和非线性最小二乘曲线拟合函数 lsqcurvefit()。非线性最小二乘法在数据拟合和参数估计等方面有广泛用途。

非线性最小二乘法及其函数 lsqnonlin()

$$\text{数学模型} \quad \min_x f(x) = \sum_i^m f_i(x)^2 + C \quad (6-14a)$$

$$lb \leq x \leq ub \quad (6-14b)$$

式中, C 为常数。

$$\text{式 (6-14a) 也可写成向量形式} \quad \min_x \frac{1}{2} \|F(x)\|_2^2 = \frac{1}{2} \sum_i^m f_i(x)^2 \quad (6-14c)$$

式中 x 为向量, 函数 $F(x)$ 为一返回向量值的函数, 即:

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} \quad (6-14d)$$

调用格式: $\mathbf{x} = \text{lsqnonlin}(@\text{fun}, \mathbf{x0})$

$[\mathbf{x}, \text{resnorm}, \text{residual}, \text{exitflag}] = \text{lsqnonlin}(@\text{fun}, \mathbf{x0}, \text{lb}, \text{ub}, \text{options}, \text{p1}, \text{p2}, \dots)$

$[\mathbf{x}, \text{resnorm}, \text{residual}, \text{exitflag}, \text{output}, \text{lambda}, \text{jacobian}] = \text{lsqnonlin}(\dots)$

输入参数: fun 定义式 (6-14d) 中的函数 $\mathbf{F}(\mathbf{x})$, 注意函数返回 $\text{fun}(\mathbf{x})$, 而不是 $\text{sum}(\text{fun}(\mathbf{X}).^2)$ 。

$\mathbf{x0}$ 初值

lb, ub 设计变量的下边界和上边界, 即 $\text{lb} \leq \mathbf{X} \leq \text{ub}$, 若无边界, 则将 lb 和 ub 赋给空矩阵: $\text{lb} = []$, $\text{ub} = []$ 。如果 $\mathbf{X}(i)$ 下无界, 则设置 $\text{lb} = -\text{inf}$; 如果 $\mathbf{X}(i)$ 上无界, 则设置 $\text{ub} = \text{inf}$;

输出参数: (同前)

非线性最小二乘曲线拟合及其函数 lsqcurvefit

数学模型: $\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{F}(\mathbf{x}, \mathbf{xdata}) - \mathbf{ydata}\|_2^2 = \frac{1}{2} \sum_i^m (\mathbf{F}(\mathbf{x}, \mathbf{xdata}_i) - \mathbf{ydata}_i)^2 \quad (6-15a)$

$\text{lb} \leq \mathbf{x} \leq \text{ub} \quad (6-15b)$

其中, \mathbf{xdata} 和 \mathbf{ydata} 为向量, $\mathbf{F}(\mathbf{x}, \mathbf{xdata})$ 为一返回向量值的函数。

函数 lsqcurvefit 的调用格式:

$\mathbf{x} = \text{lsqcurvefit}(@\text{fun}, \mathbf{x0}, \mathbf{xdata}, \mathbf{ydata})$

$\mathbf{x} = \text{lsqcurvefit}(@\text{fun}, \mathbf{x0}, \mathbf{xdata}, \mathbf{ydata}, \text{lb}, \text{ub}, \text{options}, \text{p1}, \text{p2}, \dots)$

$[\mathbf{x}, \text{resnorm}] = \text{lsqcurvefit}(@\text{fun}, \mathbf{x0}, \mathbf{xdata}, \mathbf{ydata}, \dots)$

$[\mathbf{x}, \text{resnorm}, \text{residual}, \text{exitflag}, \text{output}, \text{lambda}, \text{jacobian}] = \text{lsqcurvefit}(@\text{fun}, \mathbf{x0}, \mathbf{xdata}, \mathbf{ydata}, \dots)$

输入参数: fun 即计算向量值函数 $\mathbf{F}(\mathbf{x}, \mathbf{xdata})$ 的自定义函数, 格式为:

$\text{function } \mathbf{F} = \text{fun}(\mathbf{x}, \mathbf{xdata})$

$\mathbf{F} = \dots$ % 计算 \mathbf{x} 处的函数值向量

$\mathbf{x0}$ 初值

$\mathbf{xdata}, \mathbf{ydata}, \text{lb}, \text{ub}$ 同式 (6-15a)。

输出参数: \mathbf{x} 拟合系数向量 (参数向量)

resnorm \mathbf{x} 处的残差平方和范数值: $\text{sum}\{(\text{fun}(\mathbf{x}, \mathbf{xdata}) - \mathbf{ydata}).^2\}$

residual \mathbf{x} 处的残差值: $\text{fun}(\mathbf{x}, \mathbf{xdata}) - \mathbf{ydata}$

jacobian \mathbf{x} 处的雅可比矩阵

其他参数同前。

【例 6-11】 已知代数模型: $y = k_1 e^x + k_2$, 其中 k_1 和 k_2 为待估计参数。现令 $\mathbf{k} = [2, 6]$, 由此计算 $\mathbf{x}_i = [2; 5; 7]$ 处的 y_i , 得到的数据 (x_i, y_i) 作为“实验”数据。再利用此数据, 调用非线性最小二乘曲线拟合函数 lsqcurvefit 求取已知模型的参数 k_1 和 k_2 , 看看是否为 $\mathbf{k} = [2, 6]$ 。

程序说明 函数 Func() 定义模型表达式, 内部默认目标函数为 $\text{sum}\{(\text{Func}(x, \text{xdata}) - \text{ydata}).^2\}$ 。再以 $k_0 = [5, 5]$ 为初值, 调用 lsqcurvefit() 求解。**[注意]** 函数 lsqcurvefit() 并不要求直接定义优化目标函数, 而只定义目标函数式 (6-15a) 中的模型表达式 $F(x, \text{xdata})$ 。

程序清单 xLsqcurvefit.m

```
function xLsqcurvefit
k = [2, 6]; xdata = [2; 5; 7]; ydata = k(1)*exp(xdata) + k(2); k0 = [5, 5];
k = lsqcurvefit(@Func, k0, xdata, ydata)
% -----
function F = Func(k, xdata)
F = k(1)*exp(xdata) + k(2);
```

计算结果 $k = [2.0000 \quad 6.0000]$ 。

6.2 化工过程的设计优化

【例 6-12】冷却器的最优设计^[5]

某炼油厂需将煤油从 $T_1 = 140^\circ\text{C}$ 冷却到 $T_2 = 40^\circ\text{C}$, 煤油处理量 $G = 3 \times 10^4 \text{ kg/h}$, 冷却介质为水, 初始温度为 $t_1 = 30^\circ\text{C}$, 要求设计一冷却器 (逆流换热), 并使该冷却器的年度总费用 J 尽可能小。已知数据: (1) 冷却器每单位面积的总投资费 (材料费和制作费) $J_A = 200 \text{ 元/m}^2$ 。(2) 冷却器年折旧率 $\beta = 15\%$ (包括维修费)。(3) 冷却器总传热系数 $K = 836.8 \text{ kJ/(m}^2 \cdot \text{h} \cdot ^\circ\text{C})$ 。(4) 冷却器每年运行时间 $\theta = 8000 \text{ h}$ 。(5) 冷却水单价 $J_w = 0.04 \text{ 元/吨}$ 。(6) 冷却水比热容 $c_{pw} = 4.184 \text{ kJ/(kg} \cdot ^\circ\text{C})$ 。(7) 煤油比热容 $c_{pc} = 2.092 \text{ kJ/(kg} \cdot ^\circ\text{C})$ 。

根据传热学的基本原理分析可知, 该冷却系统存在一个最佳的冷流出口温度使总费用最低 (参见化工原理, 冷流出口温度高, 则传热推动力小, 完成规定热负荷所需的传热面积大, 投资费用就大, 而需要的冷却水量减小, 操作费用小)。试用最优化方法确定该最优冷流体出口温度, 并计算在此最优值下冷却器的最小年费用、冷却器的传热面积、每小时冷却水用量以及年度投资费和年度操作费各占总费用的百分比。

数学模型

$$(1) \text{ 目标函数} \quad J = J_A \cdot A \cdot \beta + J_w \cdot \frac{\theta \cdot w}{1000} \quad (1)$$

式中, J_A 、 β 、 J_w 、 θ 为已知参数; A 为传热面积, m^2 ; w 为冷却水用量, kg/h 。

(2) 关于 A 和 w 的数学模型——热平衡方程

$$\text{冷却器的热负荷为} \quad Q = Gc_{pc}(T_1 - T_2) \quad (2)$$

由热平衡方程 $Q = wc_{pw}(t_2 - t_1) = Gc_{pc}(T_1 - T_2) = KA\Delta t_m$, 得

$$w = \frac{Q}{c_{pw}(t_2 - t_1)} \quad (3)$$

$$A = \frac{Q}{K\Delta t_m} \quad (4)$$

式中

$$\Delta t_m = \frac{(T_1 - t_2) - (T_2 - t_1)}{\ln \frac{T_1 - t_2}{T_2 - t_1}} \quad (5)$$

由(2)和(5)代入(3)和(4),然后再代入(1),则(1)式仅有一个未知变量 t_2 。因此,该问题属于单变量最优化问题。

程序说明 采用 Nelder-Mead 单纯形法函数 `fminsearch()` 进行优化,其中 `TotalFee()` 定义目标函数式(1),函数 `Area_Water()` 则根据式(3)和(4)求 A 和 w 。

程序清单 *CoolerOptDes.m*

```
function CoolerOptDes          % 冷却器的最优化设计(Optimal Design of a Cooler)
clear all; clc
global T1 T2 G t1 JA beta K theta Jw Cw Cc Q
T1 = 140; T2 = 40; G = 2e4; t1 = 30;      % T1: °C, T2: °C, G: kg/h, t1: °C
JA = 200; beta = 0.15; K = 836.8;        % JA: yuan/m^2, K: kJ/(m^2 h °C)
theta = 8000; Jw = 0.04; Cw = 4.184;     % theta: h, Jw: yuan/ton, Cw: kJ/(kg °C)
Cc = 2.092; Q = G*Cc*(T1 - T2);         % Cc: kJ/(kg °C)
t0 = 50;                                % 给定初值
t2 = fminsearch(@TotalFee, t0);
fprintf(' 优化结果: \n\n')
fprintf('冷却器最优出口温度为: %.2f %s\n', t2, '°C')
allFee = TotalFee(t2); fprintf('最小年费用为: %.3f 元\n', allFee)
[A w] = Area_Water(t2); fprintf('冷却器传热面积为: %.3f m^2\n', A)
fprintf('每小时冷却水用量为: %.1f kg/h\n', w)
fee1 = JA*A*beta; fee2 = Jw*theta*w/1000;
fprintf('年度投资费为: %.1f 元, 占总费用: %.2f %s\n', fee1, fee1/allFee*100, '%')
fprintf('年度操作费为: %.1f 元, 占总费用: %.2f %s\n', fee2, fee2/allFee*100, '%')
% -----
function J = TotalFee(t2)
global T1 T2 G t1 JA beta K theta Jw Cw Cc Q
[A w] = Area_Water(t2); J = JA*A*beta + Jw*theta*w/1000;
% -----
function [A, w] = Area_Water(t2)
global T1 T2 G t1 JA beta K theta Jw Cw Cc Q
var1 = T1 - t2; var2 = T2 - t1; dtm = (var1 - var2)/log(var1/var2);
A = Q/(K*dtm); w = Q/Cw/(t2 - t1);
```

计算结果 冷却器最优出口温度为 92.49 °C, 最小年费用为 11352.578 元, 冷却器传热面积为 207.715 m², 每小时冷却水用量为 16003.5 kg/h。总费用中, 年度投资费为 6231.5 元, 占总费用的 54.89 %; 年度操作费为 5121.1 元, 占总费用的 45.11 %。

【例 6-13】换热器系列的最优设计^[5]

某化工厂欲利用本厂废热, 采用三个换热器将某物料温度由 100 °C 加热到 500 °C, 其流程和有关数据如图 6-1 所示, 请问各温度如何选取才能使换热器系列的总传热面积 A 为最小? 已知:

(1) 所有物料其 $m_i c_{pi} = 10^5$, 其中 m 是流体质量流率, c_p 为流体比热容, i 为流体序号, i

= 1, 2, ...;

(2) 三个换热器的总传热系数分别为 $K_1 = 120$, $K_2 = 80$, $K_3 = 40$;

(3) 为简化起见, 换热器采用逆流换热时, 其温差 Δt_m 采用算术平均值。各数值略去量纲。

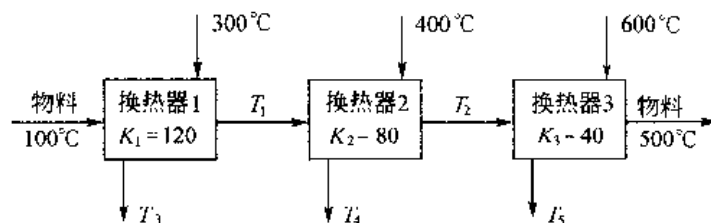


图 6-1 换热器系列

数学模型

(1) 忽略各物料热容随温度的变化, 由热量衡算得:

对换热器 1 $mc_p (T_1 - 100) = mc_p (300 - T_3)$

即 $T_3 = 400 - T_1$

对换热器 2 $mc_p (T_2 - T_1) = mc_p (400 - T_4)$

即 $T_4 = 400 - T_2 + T_1$

对换热器 3 $mc_p (500 - T_2) = mc_p (600 - T_5)$

即 $T_5 = 100 + T_2$

(2) 各换热器的温差均取算术平均值:

$$\Delta t_{m1} = [(300 - T_1) + (T_3 - 100)] / 2 = 300 - T_1$$

$$\Delta t_{m2} = [(400 - T_2) + (T_4 - T_1)] / 2 = 400 - T_2$$

$$\Delta t_{m3} = [(600 - 500) + (T_5 - T_2)] / 2 = 100$$

由热量方程 $Q = mc_p \Delta T = KA \Delta t_m$ 可得总传热面积的最小化目标函数:

$$\min A = \sum_{i=1}^3 A_i = \frac{10^5 (T_1 - 100)}{120(300 - T_1)} + \frac{10^5 (T_2 - T_1)}{80(400 - T_2)} + \frac{10^5 (500 - T_2)}{4.0 \times 100}$$

约束条件: $100 < T_1 < 300, T_1 < T_2 < 400$

程序说明 采用序贯二次规划法 (SQP) 函数 `fmincon()` 进行优化, 其中 `HeatExchangerArea()` 定义面积最小化目标函数。

程序清单 见光盘中的 `HeatExchangerDesign.m`。注意这里不能用函数 `fminsearch` 和 `fminunc`, 因为它们是无约束的优化函数, 而这里的优化问题是有约束的。

计算结果 $T_1 = 181.2504 = 181$, $T_2 = 296.2535 = 296$, 其他参数: $T_3 = 400 - T_1 = 219$, $T_4 = 400 - T_2 + T_1 = 285$, $T_5 = 100 + T_2 = 396$, $F_1 = 567.2269$, $F_2 = 1382.1922$, $F_3 = 5100$, 总面积 $F = F_1 + F_2 + F_3 = 7049.4191$ 。

【例 6-14】连续搅拌模式反应器的最优设计^[5]

在一连续搅拌槽式反应器中 (图 3-5) 进行化学反应: $2A \longrightarrow B$, 已知单位体积的液相反应速率方程为

$$-r_A = \frac{-dC_A}{dt} = 2.0C_A^2 = 2.0C_{A0}^2(1 - x_A)^2$$

式中, C_A 为反应器中物料 A 的浓度, mol/L; C_{A0} 为进料中物料 A 的浓度, mol/L; x_A 为转化率。

假定进料中 A 的浓度在一个连续的范围内变化, 且其单位成本由经验公式确定

$$c_1 = 4.0C_{A0}^{1.4} \text{ (元/升)}$$

反应器及有关辅助设备的折旧费用, 操作人员工资, 公用工程费用等假定为

$$c_2 = 0.4V_R^{0.6} \text{ (元/升)}$$

式中, V_R 为反应器体积, 为简化起见, 反应物料的体积近似等于 V 。根据预测, 当产品 B 的产量为 50mol/h 时, 就足以满足市场需求。试确定物料的进料量 F_{A0} 、浓度 C_{A0} 、反应器的体积 V 和转化率 x_A 各为多大, 使得每小时总消耗费用为最小?

即求

$$\min f = c_1 F_{A0} + c_2 = 4.0C_{A0}^{1.4} F_{A0} + 0.4V_R^{0.6}$$

数学模型 根据物料衡算 (不考虑未反应物料的再循环利用)

对反应物 A

$$F_{A0}C_{A0} = F_{A0}C_{A0}(1-x_A) - r_A V_R$$

产物 B 的生成速率

$$F_B = \frac{1}{2} F_{A0} C_{A0} x_A$$

求解以上两式得

$$V_R = \frac{F_B}{C_{A0}^2(1-x_A)^2} = \frac{50}{C_{A0}^2(1-x_A)^2}$$

$$F_{A0} = \frac{100}{C_{A0}x_A}$$

最小化目标函数为

$$\min_{C_{A0}, x_A} f = \frac{400C_{A0}^{0.4}}{x_A} + \frac{4.183}{C_{A0}^{1.2}(1-x_A)^{1.2}}$$

程序说明 采用 SQP 优化函数 fmincon() 求解, 其中 Fee() 定义目标函数, 给定初值 $[CA0, xA] = [0.5 \ 0.5]$ 。

程序清单 CSTR_OptDes.m

```
function CSTR_OptDes % 连续搅拌槽式反应器的最优设计
clear all; clc
x0 = [0.5 0.5]; % 给定初值
options = []; % 使用默认参数
xL = [0 0]; xU = [1 1]; % 设定下限 xL 和设定上限 xU
[x, f, flag] = fmincon(@Fee, x0, [], [], [], [], xL, xU)
CA0 = x(1); xA = x(2);
fprintf(' 优化结果: \n\n'), fprintf(' 转化率为: %.3f\n', xA)
fprintf(' 浓度 CA0: %.3f 摩尔/升\n', CA0)
fprintf(' 进料量 FA0 为: %.3f 升/小时\n', 100/(CA0*xA))
fprintf(' 反应器体积 V 为: %.2f 升\n', 50/CA0^2/(1-xA)^2)
fprintf(' 每小时总消耗费用: %.3f 元\n', f)
% -----
function f = Fee(x) % Objective function for Optimization
CA0 = x(1); xA = x(2);
f = 400*CA0^0.4/xA + 4.183/CA0^1.2/(1-xA)^1.2;
```

计算结果 $C_{A0} = 0.238 \text{ mol/L}$, $x_A = 0.714$, $F_{A0} = 588 \text{ L/h}$, $V = 10789 \text{ L}$, 每小时总消耗费用为 $S = 420.7 \text{ 元/小时}$ 。

6.3 化工过程的操作优化

【例 6-15】 计算间歇反应器中连串-平行复杂反应系统的最优反应温度

对于[例 4-1]的间歇反应器中的连串-平行复杂反应系统, 用最优化方法计算使产物 B 收率最大的最优反应温度。

程序说明 使产物 B 收率最大亦即使产物 B 浓度最大, 因此最优问题为 $\max_T C_B$, 即

$$\min_f f = (-C_B) \quad (1)$$

这是关于温度 T 的单变量最优化问题。用最优化方法求 C_B 的过程是: 对于给定的 T , 在 $t = 0 \sim 10^4 \text{ s}$ 的时间范围内积分模型方程 (ODEs), 得到离散数据 $C_B \sim t$, 再求出最大值 $C_{B\max} = \max(C_B)$, 式 (1) 中的 C_B 实际上就是 $C_{B\max}$ 。程序用 `fminsearch()` 搜索最优温度 T , 其中 `MassEquations()` 定义模型方程 (同例 4-1), `ObjFunc()` 定义最优目标函数式 (1)。

程序清单 见光盘中的 `TempOpt_BatchReactor.m`。

计算结果 使产物 B 收率最大的最优反应温度为 224.6°C 。

连续变量系统的最优化——庞特里阿金 (Pontryagin) 极大值原理

连续变量系统的最优化问题, 是一类最优控制问题, 其决策变量和状态变量不是离散数据, 而是在某一时间或空间域中的连续函数。

对于合成氨、合成甲醇、二氧化硫氧化、乙烯水合等可逆放热反应, 随着温度升高, 平衡常数下降。因而在管式反应器中进行这一类反应时, 常在进口区即反应混合物组成离平衡组成较远时, 采用较高的温度来加快反应速率, 而在出口区即反应混合物组成接近平衡组成时, 用较低的温度, 故在反应管的各个截面上各存在一最优温度, 从而构成了管式反应器的最优温度分布。按照这一温度分布进行反应可使产品的产量最高。在这种情况下, 决策变量温度随位置变化的函数关系称为决策变量函数, 而状态变量转化率随位置变化的函数关系称为状态变量函数。管式反应器中连续变量的最优控制问题还有许多例子, 如在磷酸钙催化剂上 α -羟基异丁酸脱氢的过程、在活性炭上使氯化氰三聚为三聚氯氰的过程以及合成醋酸乙烯酯的过程等^[6]。

间歇操作存在最优的进料速度, 它使间歇操作周期最短, 见习题 6-2, 这也是连续变量系统的最优化问题。

在生化工程方面, 也有类似的最优化问题, 如青霉素发酵优化温度曲线的确定^[10]。

$$\text{已知状态方程} \quad \frac{dx_i}{dt} = f_i(t, x_1, x_2, \dots, x_n, u_1, u_2, \dots, u_n) \quad (6-16a)$$

$$\text{I.C.} \quad t_0 = 0, \quad x_i(t_0) = x_i(0) \quad i = 1, 2, \dots, n \quad (6-16b)$$

$$\text{向量形式为} \quad \frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}) \quad (6-17a)$$

$$\text{I.C.} \quad t_0 = 0, \quad \mathbf{x}(t_0) = \mathbf{x}(0) \quad (6-17b)$$

式中, \mathbf{x} 为状态变量函数: $\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T$, \mathbf{u} 为决策变量函数: $\mathbf{u}(t) = [u_1(t), u_2(t), \dots, u_n(t)]^T$, $\mathbf{f}(t, \mathbf{x}, \mathbf{u}) = [f_1(t, \mathbf{x}, \mathbf{u}), f_2(t, \mathbf{x}, \mathbf{u}), \dots, f_n(t, \mathbf{x}, \mathbf{u})]^T$ 。

$$\text{优化的目标函数为} \quad J = \varphi(\mathbf{x}(t_f), t_f) + \int_0^{t_f} \mathbf{f}(t, \mathbf{x}, \mathbf{u}) dt \quad (6-18)$$

$$\text{最终状态 } \mathbf{x}(t_f) \text{ 满足下列约束条件} \quad \varphi(\mathbf{x}(t_f), t_f) = 0 \quad (6-19)$$

式中, $\varphi(\mathbf{x}(t_f), t_f) = [\varphi_1(\mathbf{x}(t_f), t_f), \varphi_2(\mathbf{x}(t_f), t_f), \dots, \varphi_n(\mathbf{x}(t_f), t_f)]^T$

连续变量系统的最优化问题就是, 给出式 (6-17a) ~ (6-19), 寻找最优决策 $\mathbf{u}^*(t)$, 代入 (6-17a), 从式 (6-17a)、(6-17b) 和 (6-19) 解出最优状态变量 $\mathbf{x}^*(t)$, 使目标函数式 (6-18) 最优。

庞特里阿金极大值原理引入哈密尔顿 (Hamiltonian) 函数作为辅助函数:

$$H = Y + \sum_{i=1}^n \lambda_i f_i \quad (6-20)$$

$$\text{式中 } \lambda_i \text{ 由方程} \quad \frac{d\lambda_i(t)}{dt} = -\frac{\partial Y}{\partial x_i} - \sum_{j=1}^n \lambda_j \frac{\partial f_j}{\partial x_i} \quad (6-21a)$$

与状态变量 x_i 关联起来。

$$\text{式 (6-21a) 的向量形式为} \quad \frac{d\boldsymbol{\lambda}(t)}{dt} = -\frac{\partial Y}{\partial \mathbf{x}} - \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \quad (6-21b)$$

方程 (6-21a) 或 (6-21b) 称为伴随函数 (adjoint function), 并具有下列边界条件:

$$\boldsymbol{\lambda}(t_f) = \mathbf{0} \quad (6-22)$$

庞特里阿金极大值原理是: 若要使目标函数 J 为极小 (或极大), 哈密尔顿函数 H 作为决策变量 \mathbf{u} 的函数必须是极小 (或极大), 即最优的决策变量 \mathbf{u}^* 必须满足:

$$\frac{\partial H(t, \mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}(t))}{\partial \mathbf{u}} = \mathbf{0} \quad (6-23a)$$

$$\frac{\partial H}{\partial \mathbf{u}} = \frac{\partial Y}{\partial \mathbf{u}} + \frac{\partial}{\partial \mathbf{u}} \sum_{i=1}^n \lambda_i f_i = \mathbf{0} \quad (6-23b)$$

对于管式反应器的最优温度分布问题, 状态变量为各组分的组成 x_i , 决策变量为温度 T , 其状态方程、伴随函数及其边界条件重新归纳如下:

$$\text{状态方程} \quad \begin{cases} \frac{dx_i}{dt} = f_i(t, x_1, x_2, \dots, x_n, T) \\ t_0 = 0, \quad x_i(t_0) = x_i(0), \quad i = 0, 1, \dots, n \end{cases} \quad (6-24)$$

$$\text{伴随函数} \quad \begin{cases} \frac{d\lambda_i(t)}{dt} = -\frac{\partial Y}{\partial x_i} - \sum_{j=1}^n \lambda_j \frac{\partial f_j}{\partial x_i} \\ \lambda_i(t_f) = 0, \quad i = 1, 2, \dots, n \end{cases} \quad (6-25)$$

选择决策变量 T , 使哈密尔顿函数式 (6-20) 最大, 即最优 T^* 满足:

$$\frac{\partial H}{\partial T} = \frac{\partial Y}{\partial T} + \frac{\partial}{\partial T} \sum_{i=1}^n \lambda_i f_i = 0 \quad (6-26)$$

则联立求解状态方程 (6-24)、伴随方程 (6-25) 和 (6-26) 可确定最优温度分布和最终组成。

下面通过例 6-16 介绍具体的数值解法。

【例 6-16】计算管式反应器的最优温度分布^[7]

在一柱塞流管式反应器中进行反应: $A \xrightleftharpoons[k_2]{k_1} B \xrightarrow{k_3} C$, 各反应均为一级, 反应速率常数

随温度的变化符合阿罗尼乌斯公式: $k_i = k_{i0} \exp(-\frac{E_i}{RT})$, $i = 1, 2, 3 \dots$ (1)

假设进料是纯 A, 且 $\frac{E_2}{E_1} = 2$, $k_3 = 2k_2$ 。试用庞特里阿金极大值原理计算使产品 B 的产量达到最大的最优温度分布。

数学模型

状态方程 (质量平衡方程): $\frac{dx_1}{dt} = -k_1 x_1 + k_2 x_2 = f_1$ (2a)

$$\frac{dx_2}{dt} = k_1 x_1 - (k_2 + k_3) x_2 = f_2 \quad (2b)$$

$$x_1(0) = 1.0, \quad x_2(0) = 0.0 \quad (2c)$$

式中, $t = l/u$; l 是从反应器入口到反应器内反应平面的距离; u 是反应混合物的表观速率; x_1 和 x_2 分别是组分 A 和 B 的摩尔分数。

反应速率常数由式 (1) 及已知的 $\frac{E_2}{E_1} = 2$, $k_3 = 2k_2$ 可得

$$k_1 = \exp(-\frac{E_1}{RT}) \equiv k, \quad k_2 = \exp(-\frac{E_2}{RT}) = k_1^2 = k^2, \quad k_3 = 2k^2 \quad (3)$$

$$\text{优化目标函数: } J = x_2(t_f) - x_2(0) = \int_0^{t_f} \frac{dx_2}{dt} dt = \int_0^{t_f} f_2 dt = \int_0^{t_f} Y dt \quad (4)$$

式中, $Y = f_2$ 。

按式 (6-25) 得伴随函数

$$\frac{d\lambda_1}{dt} = -\frac{\partial Y}{\partial x_1} - \sum_{j=1}^n \lambda_j \frac{\partial f_j}{\partial x_1} = -k + k(\lambda_1 - \lambda_2) \quad (5a)$$

$$\frac{d\lambda_2}{dt} = -\frac{\partial Y}{\partial x_2} - \sum_{j=1}^n \lambda_j \frac{\partial f_j}{\partial x_2} = -k^2(\lambda_1 - 3\lambda_2) + 3k^2 \quad (5b)$$

$$\lambda_1(t_f) = 0, \quad \lambda_2(t_f) = 0 \quad (5c)$$

选择决策变量 T , 使哈密尔顿函数值最大, 即满足式 (6-26):

$$\frac{\partial H}{\partial T} = \frac{\partial Y}{\partial T} + \frac{\partial}{\partial T} \sum_{i=1}^n \lambda_i f_i = 0$$

$$\text{即} \quad \frac{\partial k}{\partial T} [(1 + \lambda_2)(x_1 - 6kx_2) + \lambda_1(2kx_2 - x_1)] = 0 \quad (6)$$

令 $\bar{\lambda}_2 = \lambda_2 + 1$, 则 (5a) ~ (5c) 变为

$$\frac{d\bar{\lambda}_1}{dt} = k(\lambda_1 - \bar{\lambda}_2) \quad (7a)$$

$$\frac{d\bar{\lambda}_2}{dt} = k^2(3\bar{\lambda}_2 - \lambda_1) \quad (7b)$$

$$\lambda_1(t_f) = 0, \quad \bar{\lambda}_2(t_f) = 1 \quad (7c)$$

$$\text{由 (6) 式得到} \quad x_1(\bar{\lambda}_2 - \lambda_1) + 2kx_2(\lambda_1 - 3\bar{\lambda}_2) = 0 \quad (8)$$

联立求解状态方程 (2a) ~ (2c), 伴随方程 (7a) ~ (7c) 和 (8) 以确定最优温度分布和最终组成, 其中最优温度分布实际上就是最优 k 分布。

程序说明

由于式 (2a) ~ (2c)、(7a) ~ (7c) 和 (8) 所构成的方程组为非线性, 而且既包含 ODE 方程组, 又包含代数方程, 因此不能得到分析解。可按下列步骤进行数值求解。

① 因 k 在 $[0, 1]$ 内变化, 且在反应管入口区温度较高, 所以在 $t=0$ 时, 选取 $k=1$ 。以此为起点, 逐次假设其他点的 k 值。程序中, 设定积分位置向量 tspan1 并给定对应此向量的 k 的初值。

② 用龙格-库塔法对式 (2a) ~ 式 (2c) 从 $0 \rightarrow 1$ 积分, 计算 x_1 和 x_2 在此 tspan1 向量的上的值。

③ 用式 (7a) ~ (7c) 计算积分位置向量 tspan2 上的 λ_1 和 λ_2 , 其中向量 tspan2 是由 tspan1 的元素逆序排列得到, 即执行 MATLAB 语句“ $\text{tspan2} = \text{inverse}(\text{tspan1})$ ”。这里由于已知 $t = t_f = 1$ 时的终点边界条件, 因此直接使用龙格-库塔法从 $1 \rightarrow 0$ 积分。

④ 用式 (8) 变换得到的 k 迭代公式:
$$k^{(r+1)} = \frac{x_1(\lambda_1 - \bar{\lambda}_2)}{2x_2(\lambda_1 - 3\bar{\lambda}_2)} \quad (9)$$

(式中 r 为迭代次数) 重新计算 k 分布。

⑤ 判断是否满足收敛条件:
$$\frac{1}{N} \sum_{i=1}^N \left| \frac{k_{\text{old},i} - k_i}{k_i} \right| \leq \varepsilon \quad (10)$$

(式中, N 为向量 tspan1 的元素个数, k_i 为向量 k 的元素, $k_{\text{old},i}$ 为对应于 k_i 的上一次迭代值, ε 为设定的相对收敛精度, 程序中设定 $\varepsilon = 0.02$ 。

若满足收敛条件 (10), 则计算结束, 否则重复②~⑤的计算过程, 直至收敛为止。

以上介绍的方法, 其相应代码见程序 PFRTempOpt_A.m, 这种方法对 k 的初值要求比较苛刻, 若给定的初值与真值相差较大, 则容易发散。下面介绍另一种方法 (程序为 PFRTempOpt_B.m), 不使用迭代公式 (9), 而直接使用最优化方法, 搜索最优 k 值, 使方程 (8) 左边表达式的绝对值的平均值达到最小 (即趋近于 0), 即优化目标为

$$\min J = \frac{1}{N} \sum_i^N |x_{1,i}(\bar{\lambda}_{2,i} - \lambda_{1,i}) + 2k_i x_{2,i}(\lambda_{1,i} - 3\bar{\lambda}_{2,i})| \quad (11)$$

优化算法的基本步骤是:

① 给定 k 的初值。

② 用龙格-库塔法对式 (2a) ~ (2c) 从 $0 \rightarrow 1$ 积分, 计算 x_1 和 x_2 在向量 tspan1 上的值 (同上)。

③ 用龙格-库塔法对式 (7a) ~ (7c) 从 $1 \rightarrow 0$ 积分, 计算 λ_1 和 λ_2 在向量 tspan2 上的值 (同上)。

④ 计算式 (11) 的优化目标函数值 $J^{(r+1)}$, 其中 r 为优化算法的迭代次数。

⑤ 判定是否收敛, 即按下面式 (12) 比较此次迭代的 $J^{(r+1)}$ 与上一次迭代的 $J^{(r)}$, 若两者相对误差的绝对值满足所设定的收敛精度 ε , 则表示收敛, 否则根据优化算法的相应迭代公式计算新的 k 值 (即确定搜索方向), 重复②~⑤步骤, 直至收敛为止。

$$\left| \frac{J^{(r+1)} - J^{(r)}}{J^{(r+1)}} \right| \leq \varepsilon \quad (12)$$

显然, 不同的优化算法, 其确定搜索方向的迭代公式就不同, 这里直接调用 MATLAB 的多变量有约束最优化 (非线性规划) 函数 `fmincon()`, 不用关心其迭代公式及收敛判断工作,


```

% -----
function dxdt = ODEs(t, x, tspan, k)
k = spline(tspan, k, t); rambda(1) = x(3); rambda(2) = x(4); k1 = k; k2 = k*k; k3 = 2*k*k;
% 浓度方程
dxdt(1) = k2*x(2) - k1*x(1); dxdt(2) = k1*x(1) - (k2+k3)*x(2);
% 伴随方程
dxdt(3) = k*(rambda(1)-rambda(2)); dxdt(4) = k^2*(3*rambda(2)-rambda(1));
dxdt = dxdt';
% -----
function bc = BCfun(ya, yb, tspan, k) % 边界条件
bc = [ya(1)-1; ya(2); yb(3); yb(4)-1];

```

计算结果 由程序 PFRTempOpt_C.m 计算得到的最优分布数据如表 6-3。

表 6-3 最优分布数据

t	x_1	x_2	λ_1	λ_2	k
0.000	1.000	0.000	0.288	0.394	1.000
0.120	0.893	0.095	0.267	0.524	0.926
0.240	0.821	0.149	0.236	0.635	0.660
0.360	0.771	0.184	0.204	0.712	0.551
0.500	0.725	0.216	0.163	0.788	0.477
0.620	0.692	0.238	0.127	0.845	0.434
0.740	0.663	0.256	0.088	0.897	0.403
0.860	0.638	0.272	0.048	0.946	0.377
1.000	0.611	0.288	0.000	1.000	0.353

【例 6-17】连续回热型气-固化学热泵制冷系统的操作优化^[8]

一般而言，制冷系统的循环周期越短，系统的单位时间制冷量越大，但如果循环周期太短，吸附剂来不及吸附/解吸，会使系统单一周期内的制冷量减小，也会影响单位时间的制冷量，因此系统存在最优的循环周期。

第 4 章[例 4-18]曾经介绍过可逆气固化学热泵制冷系统的动态模拟，本例将对该制冷系统，以制冷功率最大为优化目标，确定最优循环时间及最优反应进度。

数学模型

优化目标函数

单位质量反应材料的平均制冷功率为：

$$SCP = \frac{Q_e}{t_{cyc} m_s} \quad (1)$$

$$Q_e = 7 N_s X_{opt} \Delta H_e \quad (1a)$$

式中， t_{cyc} 为循环周期； Q_e 为系统一个循环周期的制冷量； N_s 为反应盐 $SrCl_2$ 的物质的量（摩尔数），mol； X_{opt} 为优化的反应进度； ΔH_e 为氨蒸发潜热。

$$t_{cyc} = \max(t_h + t_d, t_c + t_a) + t_r \quad (1b)$$

优化目标函数为：max SCP 或 min -SCP

过程方程 过程方程及其参数完全同第4章[例4-18]，下面把过程方程重述如下。

$$\text{化学平衡方程: } \ln p_{\text{eq}} = -\frac{4983.28}{T} + 27.5100 \quad (\text{SrCl}_2/\text{NH}_3 \text{ 可逆气-固化学平衡}) \quad (2a)$$

$$\ln p_{\text{eq}} = -\frac{2764.4}{T} + 23.0269 \quad (\text{氨的气-液平衡}) \quad (2b)$$

$\text{SrCl}_2/\text{NH}_3$ 可逆气-固化学反应动力学模型:

$$\frac{dX}{dt} = k_{0a} \exp\left(-\frac{E_a}{RT}\right) \cdot [1-X]^{M_a} \cdot \frac{p_c - p_{\text{eq}}(T)}{p_{\text{eq}}(T)} \quad (\text{吸附}) \quad (3a)$$

$$\frac{dX}{dt} = k_{0d} \exp\left(-\frac{E_d}{RT}\right) \cdot [1-X]^{M_d} \cdot \frac{p_{\text{eq}}(T) - p_c}{p_{\text{eq}}(T)} \quad (\text{解吸}) \quad (3b)$$

式中, X 为反应进度; T 为床层温度; k_{0a} , E_a , M_a 及 k_{0d} , E_d , M_d 为动力学参数。

床层热量平衡方程:

$$m_s c_{ps}(X) \frac{dT_s}{dt} = h_{st}(T_t - T_s) \pm 7N_s \Delta H_r \cdot \frac{dX}{dt} \quad (4)$$

式中 c_{ps} 为单位体积床层的热容量 ($\text{J} \cdot \text{m}^3 \cdot \text{K}^{-1}$), 其计算公式如下:

$$c_{ps} = \rho_s \times w_{gr} \times c_{pgr} + c_{ps1} \times M_{s1} + c_{ps2} \times M_{s2} \quad (4a)$$

$$M_{s1} = \rho_s \times (1 - w_{gr}) \times (1 + M_g / M_s) \times (1 - r) \quad (4b)$$

$$M_{s2} = \rho_s \times (1 - w_{gr}) \times (1 + 8 \times M_g / M_s) \times r \quad (4c)$$

$r = X$ (吸附时); $r = 1 - X$ (解吸时)。

对于加热/冷却阶段, 由于不存在反应项 ($X=0$), 故床层热量平衡方程 (4) 变为:

$$m_s c_{ps} \frac{dT_s}{dt} = h_{st}(T_t - T_s) \quad (5)$$

$$\text{反应管壁的热量平衡方程: } m_t c_{pt} \frac{dT_t}{dt} = h_{st} A_{st}(T_s - T_t) - h_{tf} A_{tf}(T_t - T_f) \quad (6)$$

夹套内载热流体的热量平衡方程:

$$m_f c_{pf} \frac{dT_f}{dt} = h_{tf} A_{tf}(T_t - T_f) - h_{fe} A_{fe}(T_f - T_a) - F_f \rho_f c_{pf}(T_f - T_{fn}) \quad (7)$$

模拟计算过程

(1) 考虑上次循环回热结束时系统两个反应器的初始温度均为 T_0 , 即 $t=0$ s 时, $T_{a0} = T_{d0} = T_0$ 。其中, T_{a0} 和 T_{d0} 分别为吸附床和解吸床的初始温度, T_{fc} 和 T_{fn} 分别为冷、热油温度。

(2) 设反应进度 X

① 对于加热和冷却阶段, 先计算等容加热和冷却所需的时间, 即解吸床层从 T_{d0} 升温到 T_{ed} 所需的时间 t_h 和吸附床层从 T_{a0} 降温到 T_{ea} 所需的时间 t_c 。这可通过求解方程组 (5) ~ (7) 得到。

② 对于吸附/解吸阶段, 根据动力学方程 (3a) 和 (3b) 计算对应于 X 的吸附时间 t_a 和解吸时间 t_d , 然后求解方程组 (4)、(6)、(7) 得到对应于 t_a 和 t_d 时的各床层温度、壁温、流体 (油) 温度。

③ 对于回热阶段, 由吸附/解吸床层热量平衡方程 (形式同方程 (5))、吸附/解吸管壁热量平衡方程 (形式同方程 (6)) 和吸附/解吸器夹套内载热流体热量平衡方程 (形式同方程 (7)) 等 6 个方程构成回热微分方程组, 通过数值计算得到吸附/解吸床层、载热流体和反应管壁的温度动态变化以及所需的回热时间 t_r 。

(3) 由公式 (1b) 计算循环时间 t_{cyc} 。

(4) 由公式 (1a) 求制冷量、由公式 (1) 求平均功率 SCP 。

(5) 重复 (2) 迭代搜索直至得到最优 X 使 $(-SCP)$ 最小。此时得到最优反应进度 X_{opt} 、最优循环时间 $t_{cyc,opt}$ 、最大平均制冷功率、相应的 t_a 、 t_d 、 t_r ，以及吸附/解吸床层、载热流体和反应管壁的温度动态变化等。

本例程序从略。

计算结果 平均制冷功率随循环时间的变化曲线和一个最优循环时间($t_{cyc}=3630s$)内的吸附/解吸床层温度曲线分别示于图 6-2 (a) 和 (b)。其他结果参见文献[8]。

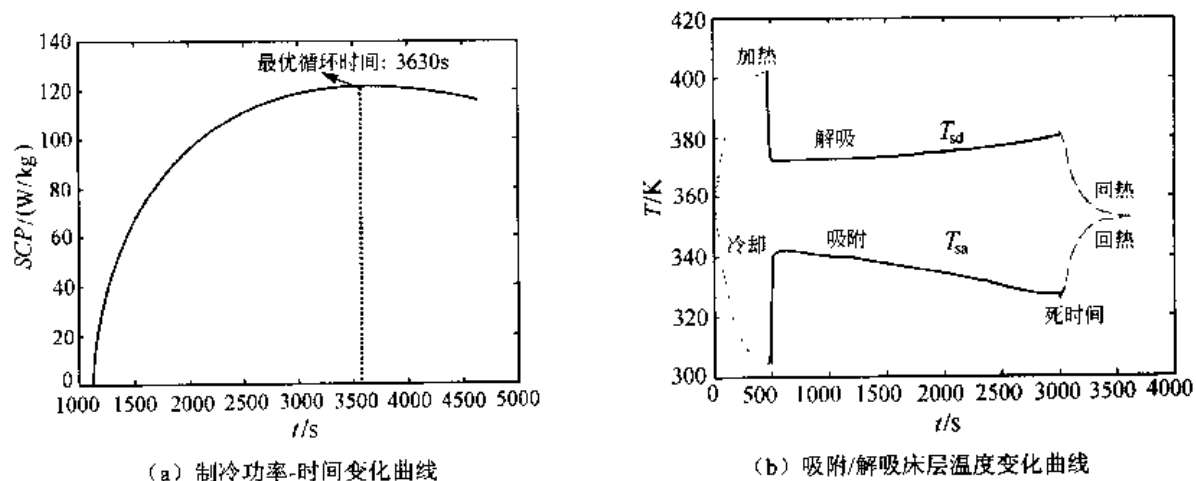


图 6-2 最优循环分布

6.4 其他化工优化问题

【例 6-18】 某化学公司用间歇法生产四种产品 A、B、C 和 D。下表中列出原料需要量、储藏场地、产量和收益。每天所用的原料总量为 18000 kg，全部产品的储藏场地为 47.5 m²。每天最多生产 7 h，每天生产结束后将全部产品装进仓库。假定四种产品必须均分全部有效储藏场地、生产时间和原料，那么为了使总收益最高，试计算每种化学产品应当生产的桶数^[9]。

产 品	A	B	C	D
原 料/(kg/桶)	200	200	150	250
储藏场地/(m ² /桶)	0.4	0.5	0.4	0.3
产 量/(桶/h)	30	60	20	30
收 益/(英镑/桶)	10	13	10	11

数学模型 设生产 A、B、C 和 D 的桶数分别为 x_1 、 x_2 、 x_3 和 x_4 ，则目标函数为

$$\max P = 10x_1 + 13x_2 + 10x_3 + 11x_4$$

$$\text{S.T. } 200x_1 + 200x_2 + 150x_3 + 250x_4 \leq 18000$$

$$0.4x_1 + 0.5x_2 + 0.4x_3 + 0.3x_4 \leq 47.5$$

$$2x_1 + x_2 + 3x_3 + 2x_4 \leq 420$$

程序说明 由模型看出，这是线性规划问题。由于优化目标函数是求最大值问题，要先把原问题转化为最小化问题（乘以-1），然后用函数 linprog() 求解。

程序清单 见光盘中的 LinearProg.m。

计算结果 $x = [0 \ 15 \ 100 \ 0]$, 即为获得最优收益, 每天应生产 15 桶 B 和 100 桶 C。将 x 优化值代入三个约束条件可知, 每天生产 15 桶 B 和 100 桶 C (不生产 A 和 D) 正好用去全部原料和全部储存场地, 而每天还剩余 105min 的生产能力。

6.5 全局最优化简介

如果非线性 (NLP) 算法, 如 SLP、SQP 或 GRG 应用于光滑非凸问题 (smooth nonconvex problem), 通常收敛于最近的局部极小值, 但可能不是全局最小。我们把这些算法称为“局部求解器 (local solvers)”。前面介绍的方法都是局部最优化方法。MATLAB 最优化工具箱的目前 6.5 版本仅提供局部最优化函数, 将来的版本可能提供全局最优化算法函数。

全局最优化方法的分类可参见 Edgar 等^[1]。下面仅介绍几种最常见的全局最优化方法。

(1) multistart 方法 从多个起始点开始调用局部求解器。具体地说, 给定一个起始点, 调用 SLP、SQP 或 GRG 这些局部优化算法, 求得局部最优解, 然后改变新的起始点, 重复调用这些局部求解器, 得到新的局部最优解, 这样不断重复进行, 直到目标函数值最小为止。由于在 6.1 节中已经介绍了 MATLAB 的局部优化算法, 因此, 我们已经能够用这个方法求解全局最优值。

(2) 遗传算法 (Genetic Algorithm, GA)^[11] 遗传算法是模拟生物体内染色体群体的遗传进化过程的一种高效搜索算法, 它是美国学者 Holland 于 1975 年首先提出来的。它摒弃了传统的搜索方式, 模拟自然界生物进化过程, 采用人工进化的方式对目标空间进行随机化搜索 (概率搜索)。它将问题域中的可能解看作是群体的一个个体或染色体, 并将每一个体编码成符号串形式, 模拟达尔文的遗传选择和自然淘汰的生物进化过程, 对群体反复进行基于遗传学的操作 (遗传, 交叉和变异), 根据预定的目标适应度函数对每个个体进行评价, 依据适者生存, 优胜劣汰的进化规则, 不断得到更优的群体, 同时以全局并行搜索方式来搜索优化群体中的最优个体, 求得满足要求的最优解。

(3) 模拟退火算法 (Simulated Annealing, SA)^[12] 它模拟了固体退火过程中系统内原子群体趋向能量最低的基态的过程。

(4) 蚁群算法 (Ant Colony)^[13] 蚂蚁算法是近几年问世并逐步引起重视的一种新的全局优化仿生算法, 它模拟了蚁群通过个体之间的信息交流与相互协作找到蚁穴到食物源的最短路径的过程, 是一种通用型随机优化方法。

遗传算法、模拟退火算法和蚁群算法都是通过模拟自然领域中具有某种优化特征的群体演化过程而建立起来的新型而高效的全局优化搜索算法。它们已成功的解决了许多复杂的优化问题。

习 题

6-1 用庞特里阿金极大值原理计算管式反应器的最优温度分布

在一管式反应器内进行化学反应: $A \xrightarrow{k_1} B \xrightarrow{k_2} C$, 所有的反应均为一级, 且反应速率常数随温度的变化符合阿罗尼乌斯公式 $k_i = k_{i0} \exp(-\frac{E_i}{RT})$, 式中 $k_{10} = 0.535 \times 10^{11} \text{ min}^{-1}$, $k_{20} = 0.461 \times 10^{18} \text{ min}^{-1}$, $E_1 = 1.8 \times 10^4 \text{ kcal/kmol}$, $E_2 = 3.0 \times 10^4 \text{ kcal/kmol}$ 。假设通过反应器的流动是柱塞流, 为使产品 B 的产量达到最大, 推导计算温度分布的方程组。若初始料液混合物中 A 的浓度为 0.95 kmol/m^3 , B 的浓度为 0.05 kmol/m^3 , 用庞特里阿金极大值原理计算最优温度分布。(文献[9]引自 Lee E. S., Quasilinearisation and Invariant Imbedding. Academic Press, New York, 1968.)

6-2 用庞特里阿金极大值原理计算间歇过程的最优进料速度^[7]

如图 6-3 所示流程, 原料经外部的预热器用蒸汽加热后进入一个间歇操作的容器, 其底部有一传热面积很小的蛇管换热器对进入的物料进一步加热到所需温度 T_2 , 并在以后维持容器内反应的进行。在这种情况下, 如加料速度太快, 通过外部换热器的预热温度降低, 必须由蛇管从较低温度继续加热, 使间歇操作的周期加长。若加料速度很小, 因加料时间过长, 同样也会延长操作周期。因而存在着最优的进料速度, 它使间歇操作周期最短。

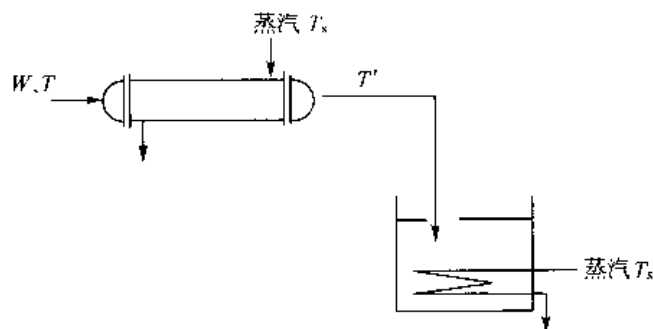


图 6-3 间歇操作流程

已知数学模型如下:

$$\text{在加料期间, 容器中物料的积累速率为} \quad \frac{dM}{dt} = W \quad (1)$$

式中, M 为物料量; t 为时间, W 为加料速率。

$$\text{在加料期间任意时刻,} \quad Wc_p(T' - T_c) = UA_1(T' - T_c) - \frac{1}{\ln \frac{T_s - T_c}{T_s - T'}}$$

$$\text{即} \quad T' = T_s - (T_s - T_c) / \exp\left(\frac{UA_1}{Wc_p}\right) \quad (2)$$

式中, c_p 为比热容; T 为换热器进口处的原料温度; T_s 为水蒸气温度, UA_1 为换热器传热面积与传热系数的乘积。

$$\text{在加料期间对容器的热量衡算为:} \quad \frac{d(Mc_p T)}{dt} = Wc_p T' + UA_2(T_s - T) \quad (3)$$

将上式的左侧展开并与式 (1)、(2) 相结合, 得到

$$\frac{dT}{dt} = \left(\frac{W}{M} + \frac{UA_2}{Mc_p} \right) (T_s - T) - \frac{W(T_s - T_c)}{M \exp\left(\frac{UA_1}{Wc_p}\right)} \quad (4)$$

因此, 得到状态方程 (1) 和 (4), 其中 M 和 T 为状态变量, W 为控制变量 (决策变量)。

已知状态参数如表 6-4。

表 6-4 状态参数

$UA_1/[\text{kcal}/(\text{h} \cdot ^\circ\text{C})]$	$UA_2/[\text{kcal}/(\text{h} \cdot ^\circ\text{C})]$	$c_p/[\text{kcal}/(\text{kg} \cdot ^\circ\text{C})]$	M_0/kg	$T_0/^\circ\text{C}$	$T_0/^\circ\text{C}$	$T_s/^\circ\text{C}$
10	10	1	10	0	25	100

参 考 文 献

- 1 Edgar, T. F., Himmelblau D. M. and Lasdon L. S., Optimization of Chemical Processes. 2nd edition. McGraw-Hill, 2001
- 2 Optimization Toolbox User's Guide. Version 2.2. The MathWorks, Inc., 2002 (optim_tb.pdf)
- 3 杨冰. 实用最优化方法及计算机程序. 哈尔滨: 哈尔滨船舶工程学院出版社, 1994

- 4 苏金明, 阮沈勇编著. MATLAB 6.1 实用指南 (下册). 北京: 电子工业出版社, 2002
- 5 邓正龙主编. 化工中的优化方法. 北京: 化学工业出版社, 1992: 21, 29, 40~43, 75
- 6 Ю.И. 马特洛斯著, 李成岳, 阎泽群译. 催化反应器中的非定态过程. 北京: 科学出版社, 1994
- 7 张建侯, 许锡恩编著. 化工过程分析与计算机模拟. 北京: 化学工业出版社, 1989, 400~402, 454~458
- 8 黄华江, 戴迎春, 袁渭康. 连续回热型气-固化学热泵制冷系统的操作优化. 化学反应工程与工艺, 2003(3)
- 9 V. G. Jenson and G. V., Mathematical Methods in Chemical Engineering, 2nd edition, Academic Press, London, 1977. 中译本, [英] V.G.詹森, G.V.杰弗里斯, 台德荣译. 化工数学方法. 第二版. 北京: 化学工业出版社, 1982, 506
- 10 Constantinides A. and Navid Mostoufi, Numerical Methods for Chemical Engineers with MATLAB Application, Prentice Hall, Upper Saddle River, NJ, 1999. 331. From: Constantinides, A., Spencer, J. L., and Gaden, E. L., Jr., "Optimization of Batch Fermentation Processes, I. Development of Mathematical Models for Batch Penicillin Fermentations," Biotech. Bioeng., vol. 12, 1970, 803
- 11 Holland J H. Genetic Algorithms [J]. Scientific American. 1992, 267(1). 44~50
- 12 Kirkpatrick S, Gelatt C D, Vecchi Jr M P. Optimization by simulated annealing [J]. Science, 1983, 220(4 598). 671~680
- 13 Marco Dorigo, Vittorio Maniezzo, Alberto Colomi. Ant colony system: Optimization by a colony of cooperating agents [J]. IEEE Trans on System. Man and Cybernetic, 1996, 26(1): 29~41

第7章 参数估计和模型辨识

7.1 概述

在化学工程、生物化工和环境工程等工程科学领域,通常根据系统或过程中的物理化学原理、热力学、动力学规律和传递现象等建立数学模型,以便进行模拟计算和过程分析。由于建立的数学模型常常包含未知参数,在模型化过程中,首先通过合理地(或最优地)设计试验,得到包含过程重要(或最大)信息在内的试验数据,然后利用最优化方法和统计方法进行参数估计(parameter estimation)以获得未知参数的估计值。参数估计又称数据回归(data regression)或回归分析(analysis of regression)。参数估计本质上是最优化问题,即搜索未知参数的最优值,使得模型计算值与试验测量值尽量一致。

此外,对于新探索的系统或过程,有时我们对模型结构还不确定,通过一些机理和假设可以得到两个以上的数学模型。因此,需要通过合理安排试验得到满意的试验数据,并利用统计理论进行分析,从多个模型中辨识(或筛选)出最合适的一个模型,这就是模型辨识(model identification/discrimination)过程。

简言之,过程模型化需要解决以下问题:

- (1) 确定模型的参数——参数估计;
- (2) 对模型进行判别,以便从多种模型中筛选出最佳模型——模型辨识;
- (3) 合理地设计实验,以便较好地确定模型参数和对竞争模型进行有效的判别——实验设计。

实际上,参数估计、模型辨识和实验设计通常不是孤立的,而是交互的。但是,为便于叙述,本章假设实验数据已有,着重介绍参数估计和模型辨识的具体方法,而有关试验设计的内容将在第8章中介绍。与前面各章类似,本章仍将给出足够的实例以强调实际应用。

7.2 数学模型结构

在科学与工程计算中,含有待估计参数的数学模型结构主要有两大类:

- (1) 代数方程模型——代数方程(组);
- (2) 微分方程模型——包括常微分方程(组)和偏微分方程(组)。

7.2.1 代数方程模型

代数方程模型的一般形式为: $\mathbf{y} = \mathbf{f}(\mathbf{x}, \boldsymbol{\beta})$ (7-1)

式中, \mathbf{x} 为 n 维自变量(回归变量或输入变量)向量: $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$

$\boldsymbol{\beta}$ 为 p 维参数向量: $\boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_p]^T$

\mathbf{y} 为 m 维因变量(响应变量或输出变量)向量: $\mathbf{y} = [y_1, y_2, \dots, y_m]^T$

\mathbf{f} 为模型方程的 m 维函数向量: $\mathbf{f} = [f_1, f_2, \dots, f_m]^T$

乙烯加氢反应产生乙烷的反应速率方程是代数方程模型的一个例子,该模型如下:

$$-r_A = \frac{k p_E p_H}{1 + K_E p_E}$$

式中待估参数向量为: $\beta = [k, K_E]^T$ 。

一般地, 测量的输出向量可看成是由模型方程 (7-1) 计算得到的确定性部分与随机误差部分之和:

$$\hat{y}_i = f(x, \beta) + \varepsilon_i \quad i = 1, 2, \dots, m \quad (7-2)$$

若自变量和响应变量都有实验误差, 实验数据可表示为 $\{(\hat{x}_i, \hat{y}_i), i = 1, 2, \dots, N\}$, 则有:

$$\hat{x}_i = x_i + \varepsilon_{xi} \quad (7-3)$$

自变量的误差一般比较小, 为简便起见, 通常只考虑响应变量的实验误差。

7.2.2 微分方程模型

7.2.2.1 ODE 方程模型

常微分方程模型的一般形式为:

$$\frac{dx}{dt} = f(t, x, \beta, u), \quad x(t_0) = x_0 \quad (7-4)$$

式中, β —— p 维待估计 (辨识) 的参数向量, $\beta = [\beta_1, \beta_2, \dots, \beta_p]^T$;

u —— r 维操作变量向量, $u = [u_1, u_2, \dots, u_r]^T$, 该变量向量由实验设定或实验测量得到;

x —— n 维状态变量向量, $x = [x_1, x_2, \dots, x_n]^T$;

x_0 —— n 维状态变量初始条件向量, $x_0 = [x_{10}, x_{20}, \dots, x_{n0}]^T$;

f ——已知 ODE 模型方程的 m 维函数向量, $f = [f_1, f_2, \dots, f_m]^T$ 。

由于 u 通常已知, 式 (7-4) 也可简化为

$$\frac{dx}{dt} = f(t, x, \beta), \quad x(t_0) = x_0 \quad (7-5)$$

若给定 β 值, 则可用龙格-库塔法积分式 (7-5) 得到 $x(t)$ 。因此, 输出向量可表示为:

$$y(t) = g(x(t), \beta) \quad (7-6)$$

式中, y 为 m 维输出向量, 即实验测量的一组变量, $y = [y_1, y_2, \dots, y_m]^T$, 它代表测量的状态变量的子集或者它们的组合, 也有一些状态变量亦即输出变量的特例: $y = x$ 。

g 为已知的输出向量与状态向量之间非线性关系的 m 维函数向量: $g = [g_1, g_2, \dots, g_m]^T$

与代数模型相似, 在时间 t_i 处测量的输出向量可看成是由模型方程 (7-6) (用真实参数值) 计算得到的确定性部分与随机误差部分之和:

$$\hat{y}_i = y(t_i) + \varepsilon_i \quad i = 1, 2, \dots, m \quad (7-7)$$

在动态系统中, 我们有时做了许多套实验, 这时, 通常希望同时使用所有的实验数据来估计参数。

通常, 整个状态向量 $x(t)$ 被测量得到, 因此, 可直接近似求出对状态变量的时间导数, 即式 (7-5) 的等号左侧导数项 dx/dt , 这样式 (7-5) 变为代数方程组, 由此可求出输出向量 $y(t)$; 也可利用适当的积分方法, 由式 (7-5) 求得输出向量 $y(t)$ 。利用这两种方法——导数法 (the derivative approach) 和积分法 (the integral approach), 使 ODE 系统的参数估计问题降为代数方程组的参数估计问题。

7.2.2.2 PDE 方程模型

PDE 模型参数估计要比 ODE 困难, 若参数估计直接调用 MATLAB PDE 工具箱求解 PDE 模型, 也是可以的, 但由于优化过程中每迭代搜索一次, 即求解一次 PDE 问题, 速度可能会很慢。因此, 通常对空间导数进行适当离散 (如 MOL 法) 使 PDE 方程转变成 ODE 方程组, 然后利用 ODE 模型参数估计方法进行估计。

7.3 参数估计

7.3.1 一般参数估计方法

7.3.1.1 优化准则

参数估计问题实际上是最优化问题，而最优化目标的确定是很重要的一个步骤。

用于参数估计的优化技术通常需要一个有效的误差统计准则（如最小平方和）。使用合适的误差准则，如最小二乘、极大似然或概率密度函数等，可估计得到系统参数的最优值。

对于大多数非线性参数估计问题，优化准则采用最小二乘法可得到满意的最优解。为此，本书有关最优参数估计问题，均基于最小二乘准则。基于极大似然准则或概率密度函数的最优化技术可参考其他文献。

7.3.1.2 多元线性回归

多元线性回归可用于估计线性模型的参数。此外，对非线性模型估计参数时，参数初值的选取是很关键的，这些参数初值有时可用多元线性回归获得，即先将非线性模型变换为线性模型，再用多元线性回归对该线性模型估计其参数值，以此作为非线性参数估计的初值。所以，多元线性回归仍广泛应用。

多元线性回归的回归模型为 $y = b_0 + b_1x_{i1} + b_2x_{i2} + \cdots + b_nx_{in} \quad i = 1, 2, \cdots, n$ (7-8)

模型中各系数和常数项通常利用最小二乘法求得，相应的 MATLAB 函数是 regress()。

函数 regress() 的用法

功能：用最小二乘法进行多元线性回归，即确定下列线性模型中的参数 β ：

$$y = X\beta + \varepsilon \quad (7-9)$$

$$\varepsilon \sim N(0, \sigma^2 I) \quad (7-10)$$

式中， y 为 $n \times 1$ 的观测向量； X 为 $n \times p$ 的矩阵； β 为 $p \times 1$ 的参数向量； ε 为 $n \times 1$ 的随机分布向量。

调用格式： `b = regress(y, X)`

`[b, bint, r, rint, stats] = regress(y, X, alpha)`

输入参数：

y, X 同式 (7-9) 中的 y, X

α 置信水平，即用于计算参数向量 b 和残差向量 r 的 $100(1-\alpha)\%$ 置信区间(分别用 $bint$ 、 $rint$ 表示)。默认 $\alpha = 0.05$ ，即 95% 置信区间。

输出参数：

b 估计的参数向量，即式 (7-9) 中 β 的估计值。

$bint$ 参数向量 b 的 $100(1-\alpha)\%$ 置信区间

r 残差向量

$rint$ 残差向量 r 的 $100(1-\alpha)\%$ 置信区间

$stats$ 该向量包含 R^2 统计量（相关系数平方值）、回归的 F 值和 p 值。

注 X 矩阵应该包含元素全为 1 的一列，从而使模型包含一个常数项。 F 值和 p 值是在假设模型包含一个常数项的情况下进行计算的，若模型不包含常数项，则 F 值和 p 值不正确。 R^2 值是回归残差平方和与总的残差平方和之比。

简单地，用下面的 MATLAB 命令

`>>b = X\y`

也可以实现与“>>b = regress(y, X)”相同的功能。

7.3.1.3 多响应非线性回归——多响应加权最小二乘法

一般来说，化工数学模型是参数非线性的，因此这里主要叙述非线性系统。无论是代数模型（7-1），还是微分方程模型，如（7-5）和（7-6），参数估计方法实际上是一样的，都可以表示为下列的最优化问题：

$$\min_{\beta} S(\beta) = \sum_{j=1}^v w_j [\hat{y}_j - y_j(t, \beta)]^T [\hat{y}_j - y_j(t, \beta)] \quad (7-11)$$

式中， β 是待估计参数， \hat{y}_j 是输出（响应）变量 j 在所有时间 t_i （时间向量 t ）的测量值向量， $y_j(t, \beta)$ 是根据已知数学模型计算得到在时间向量 t 的模拟值（与 \hat{y}_j 相对应）， w_j 是根据输出向量实验误差的统计特性而确定的权值。

对于权值全部为 1 的场合，（7-11）式变为普通的最小二乘法。此外，只有单个响应变量的系统，其优化目标是式（7-11）的特例，因此，多响应加权最小二乘法具有通用性。

下面以等温积分反应器的动力学参数估计为例^[1]，来说明多响应加权最小二乘法：

设在一等温积分反应器的动力学实验中，发生反应： $A \xrightarrow{k_1} B \xrightarrow{k_2} C$ ，已知当 $t = 0$ 时， $C_{A0} = 1$ ， $C_{B0} = 0$ ， $C_{C0} = 0$ 。实验数据如下表。

i	1	2	3	4	5	6	7	8	9
t_i	0	5	10	15	20	50	80	100	200
$C_{Ai,exp}$	1.000	0.840	0.679	0.503	0.425	0.137	0.030	0.006	0.000
$C_{Bi,exp}$	0.000	0.155	0.327	0.426	0.477	0.571	0.502	0.385	0.145

当 $t = 15$ 时，重复实验所得结果如下。

$C_{Ai,exp}$	0.476	0.488	0.548	0.495	0.508
$C_{Bi,exp}$	0.442	0.419	0.426	0.409	0.432

请用多响应加权最小二乘法估计参数 k_1 和 k_2 。

对此例的估计方法如下：

由于 $C_A + C_B + C_C = 1.0$ ，所以描述反应器出口反应物浓度变化的模型方程只需两个：

$$C_A = e^{-k_1 t}, \quad C_B = \frac{k_1}{k_2 - k_1} (e^{-k_1 t} - e^{-k_2 t})$$

使用最小二乘准则，即使 C_A 和 C_B 的模拟值与实验值之残差平方和（ S_A 和 S_B ）最小，即

$$\min S_A = \sum_{i=1}^9 (C_{Ai, sim} - C_{Ai, exp})^2, \quad \min S_B = \sum_{i=1}^9 (C_{Bi, sim} - C_{Bi, exp})^2$$

式中， $C_{Ai, sim}$ 和 $C_{Bi, sim}$ 分别是 A 和 B 对应于 t_i 时刻的模拟值（根据模型方程计算）。

由于反应速度常数 k_1 、 k_2 是非独立的，因此应将以上两式合并，并考虑各自的实验误差的不同权重，即对合并后的目标函数式中各项进行适当加权，以便使模型对各观测值的残差效应与各自的实验误差相一致：

$$\min w_1 \sum_{i=1}^9 (C_{Ai, sim} - C_{Ai, exp})^2 + w_2 \sum_{i=1}^9 (C_{Bi, sim} - C_{Bi, exp})^2$$

式中 w_1 和 w_2 分别对应于 C_A 和 C_B 两项的权。这就是多响应加权最小二乘问题的目标函数，利用最优化方法，可以求取模型方程中的参数 k_1 、 k_2 ，使该目标函数值最小。

根据以上 5 组重复实验得到的 C_A 和 C_B 数据的误差分析, 可算出目标函数中第一项的权值为 $w_1 = 1.0$, 第二项的权值为 $w_2 = 4.9$, 计算公式见式 (7-12)。

多响应加权最小二乘非线性参数估计的具体步骤:

① 给定参数 β 的初值;

② 根据已知的代数模型 (7-1) 或者微分方程模型, 如式 (7-5), 计算在对应时间向量 t 的模拟值 $y_f(t, \beta)$, 该值与实验值 y_j 相对应。

对于代数模型, 若是关于 y 的显函数, 可容易直接求出模拟值 $y_f(t, \beta)$; 若是关于 y 的隐函数, 则可通过求解非线性代数方程得到模拟值 $y_f(t, \beta)$ 。对于 ODE 方程模型, 如式 (7-5), 可用龙格-库塔法积分得到 $x_j(t)$, 再由式 (7-6) 求得 $y_f(t, \beta)$; 对于 PDE 方程模型, 可先用 MOL 法变换为 ODE 方程组, 然后用上面介绍的方法求得 $y_f(t, \beta)$ 。

③ 根据输出向量实验误差的统计特性确定权值 w_j ($j = 1, 2, \dots, v$)。权值公式为:

$$w_j = \frac{1/\sigma_j^2}{\sum_{i=1}^v \frac{1}{n_i} \left[\sum_{l=1}^v \sum_{i=1}^{n_i} \frac{1}{\sigma_i^2} \right]} = \frac{\sum_{i=1}^v n_i}{\sigma_j^2 \left[\sum_{i=1}^v \sum_{l=1}^{n_i} \frac{1}{\sigma_i^2} \right]} \quad (7-12)$$

式中, σ_j^2 或 σ_i^2 = 每条曲线的方差; n_i = 每条曲线的实验点数; v = 要拟合的变量数。

④ 计算式 (7-11) 的优化目标函数值, 即按式 (7-13) 计算 $S^{(r+1)}$:

$$S^{(r+1)} = \sum_{j=1}^v w_j [\hat{y}_j - y_j(t, \beta)]^T [\hat{y}_j - y_j(t, \beta)] \quad (7-13)$$

其中 r 为优化算法的迭代次数。

⑤ 判定是否收敛, 即按下面式 (7-14) 比较此次迭代的 $S^{(r+1)}$ 与上一次迭代的 $S^{(r)}$, 若两者相对误差的绝对值满足所设定的收敛精度 ϵ , 则表示收敛, 否则根据优化算法的相应迭代公式计算新的 β 值 (即确定搜索方向), 重复②~⑤步骤, 直至收敛为止, 此时可认为目标函数值最小。

$$\left| \frac{S^{(r+1)} - S^{(r)}}{S^{(r+1)}} \right| \leq \epsilon \quad (7-14)$$

可用各种优化方法搜索最优参数 β , 使式 (7-11) 的目标函数值最小。不同优化算法, 其确定搜索方向的迭代公式及收敛速度也不同。考虑到式 (7-11) 是平方和形式的函数最小化问题 (即最小二乘问题), 应该使用具有二次收敛算法的高斯-牛顿法 (Gauss-Newton Method) 和 Levenberg-Marquart 法以加快速度 (参见 7.1.8)。尤其是对于微分方程模型, 由于每次迭代计算目标函数需要积分微分方程, 比较耗时, 所以更应使用二次收敛算法。

要了解参数估计的细节, 可参考文献[2~5]。这里直接使用上一章介绍的 MATLAB 非线性最小二乘法函数 `lsqnonlin()` 和非线性最小二乘曲线拟合函数 `lsqcurvefit()`, 不用关心其迭代公式及收敛判断工作, 只需定义优化目标函数, 并给出决策变量 β 的初值, 即可调用这些函数得到最优解。需注意的是, 以上②和③步的计算过程应放在自定义的优化目标函数内。

下面再介绍函数 `nlinfit()`、`nlparci()`、`nlpredci()` 和 `nlintool()`, 其中 `nlinfit()` 是函数 `lsqnonlin()` 和 `lsqcurvefit()` 的简单版本, 而 `nlparci()` 和 `nlpredci()` 分别用于计算参数和预测值的置信区间。

函数 `nlinfit()`

功能: 用 Gauss-Newton 法进行非线性最小二乘数据拟合。

调用格式: `[beta, r, j] = nlinfit(X, y, @fun, beta0)`

输入参数: `X` 向量或矩阵, 行数与 `y` 相同;
`y` 向量;
`fun` 定义非线性函数: $\hat{y} = f(\beta, X)$, 返回给定参数初值和自变量 `X` 下的预测值;
`beta0` 参数初值。

输出参数: `beta` 拟合 (估计) 得到的参数;
`r` 残差;
`j` 雅可比 (Jacobian) 矩阵。

利用输出参数 `beta`、`r` 和 `j`, 可分别调用函数 `nlparci()` 和 `nlpredci()` 来计算参数和预测值的置信区间。

函数 `nlparci()`

功能: 计算非线性模型中参数估计值的置信区间。

调用格式: `ci = nlparci(beta, resid, j)`

输入参数: `beta`、`r` 和 `j` 即函数 `nlinfit()` 的输出结果。

输出参数: `ci` 最小二乘参数估计 `beta` 的 95% 的置信区间。

函数 `nlpredci()`

功能: 计算非线性模型中预测值的置信区间。

调用格式: `[ypred, delta] = nlpredci(@fun, input, beta, resid, j)`

`[ypred, delta] = nlpredci(@fun, x, beta, resid, j, alpha, simopt, predopt)`

输入参数: `fun`、`beta`、`resid` 和 `j` 同 `nlinfit()`;

`input` 非线性函数中自变量的数值矩阵;

`alpha` 指定置信水平为 $100(1-\alpha)\%$, 其默认值为 $\alpha = 0.05$;

`simopt` 设为 'on' 时, 为同步置信区间; 设为 'off' (默认选项) 时, 为异步置信区间;

`predopt` 对于输入处的函数置信区间, 设为 'curve' (默认选项), 对于新的响应值, 则设为 'observation'。

输出参数: `ypred` 在输入 `input` 处的预测值;

`delta` 在输入 `input` 处函数置信水平 $100(1-\alpha)\%$ 的半宽度, 即在指定输入值 `input` 处函数真值在置信水平为 $100(1-\alpha)\%$ 时的置信区间为 $[ypred - delta, ypred + delta]$ 。

在上述三个函数中, 通常先用 `nlinfit()` 计算参数 `beta`、残差 `r` 和雅可比矩阵 `j`, 然后分别调用函数 `nlparci()` 和 `nlpredci()`, 来分别计算参数和预测值的置信区间。

当然, 也可以用第 6 章介绍的函数 `lsqnonlin()` 或 `lsqcurvefit()` 代替 `nlinfit()` 以计算 `beta`、残差 `r` 和雅可比矩阵 `j`, 然后调用 `nlparci()` 和 `nlpredci()` 来分别计算参数和预测值的置信区间。

函数 `nlintool()`

功能: 用 Gauss-Newton 法进行非线性最小二乘数据拟合, 并显示交互图形。`nlintool()` 实际上把以上函数 `nlinfit()`、`nlparci()` 和 `nlpredci()` 等集成在一起, 通过图形用户界面解决非线性最小二乘数据拟合的问题。

调用格式: `nlintool(x, y, @fun, beta0, alpha)`

`nlintool(x, y, @fun, beta0, alpha, 'xname', 'yname')`

其中, 字符串'xname', 'yname'分别用于标注图形的 x 变量和 y 变量, 其他输入参数同前。

7.3.1.4 参数估计的几个重要方面

对非线性模型, 必须使用非线性回归方法:

对于非线性参数估计, 有时可以对代数模型方程进行重排, 使非线性模型转变为线性模型, 然后用线性回归法估算模型参数。以前由于计算技术的限制, 许多科学与工程人员用这种方法估计非线性模型参数。但是, 这样做实际上是违反统计分析的某些前提的(或者说这在统计上是不严格的, 往往得到错误的参数估计的置信区间)。此外, 随着计算技术的发展, 非线性拟合问题越来越容易求解。因此, 建议采用非线性参数拟合(估计)方法求取最终的模型参数。当然, 如前所述, 将非线性问题化为线性问题, 再用多元线性回归拟合得到参数值, 然后以此作为非线性参数估计的初值, 用非线性参数估计方法得到最终的参数值。

参数估计应考虑的内容:

① 对被估参数进行约束通常是重要的(比如, 参数只能为正)。有时根据先验知识, 可确定参数的大致范围(下限和上限);

② 由于数值的不精确性, 如果数值数量级相差很大, 参数和数据放大可能是必要的;

③ 绘制模型与测量值之间的残差曲线(分析图), 对识别系统或模型误差很有用。残差分析图的绘制: 将各次实验值 y_{ei} 对相应的模型计算值 y_{ci} 标绘, 检查是否均匀地分布在对角线两侧。或者将各次实验的残差 $(y_{ei} - y_{ci})$ 对 y_{ci} 标绘, 检查是否均匀地分布在零线的上下侧;

④ 序贯实验法能节省大量实验, 大大提高模型的适定性和参数估计精度。

参数估计实际指南^[3]

(1) 先检查数据

- 视觉检查 由数据绘制图形, 以直观检查数据质量和找出 outlier 数据, 这是很重要的。
- 统计方法 如果一个残差比标准差大 3~4 倍, 则它可视为 outlier, 因为此数据出现的概率只有 0.1%。

工程数据分析的一般方法:

① 把数据图表化, 以便仔细观察;

② 辨认出第 1 组可疑数据 (potential outlier);

③ 根据统计理由确认是否抛弃这些可疑数据;

④ 估计参数, 并获得响应变量值、残差等;

⑤ 绘制残差图, 检查是否存在超过 3 倍标准差的数据点;

⑥ 辨认出第 2 组可疑数据后, 检查是否由于非统计的理由应该丢弃这些 outliers, 并从数据集中删除掉;

⑦ 对余下的 outliers 数据检查它们对模型响应及估计的参数值的影响, 即进行两次参数估计计算: 一次是用包含 outliers 数据, 一次是用不包含 outliers 数据;

⑧ 检查 outliers 中哪一个比余下数据点提供更多的信息, 余下的 outliers 对模型参数或模型的平均估计响应的影响应该很小, 因此可以忽略;

⑨ 如果时间、金钱及物质环境允许的话, 在所发现 outliers 的实验条件下重复进行实验, 以补充抛弃的实验点。

(2) 初始估计值(猜测值)的确定

① 根据模型的结构和本质

描述物理系统的模型结构和本质,可能指明未知参数的可接受范围的值。此外,用不同参数值重复计算响应变量并把结果绘图,可帮助了解模型的行为及其对参数的依赖性。

② 模型方程的渐进行为

例如,考虑酶催化反应的 Michaelis-Menten 动力学: $y_i = \frac{k_1 x_i}{k_2 + x_i}$

当 $x_i \rightarrow 0$ 时,动力学方程变为 $y_i = \frac{k_1}{k_2} x_i$

因此,从最初 x_i 很小的实验数据,可根据上式 y_i-x_i 线性关系的斜率获得 k_1/k_2 。

此外,当 $x_i \rightarrow \infty$ 时,动力学方程变为 $y_i \approx k_1$ 。

因此,在很大的 x_i 处可获得 y_i 的渐进值。

又如,在分析环保样品中常碰到的指数衰变模型 (exponential decay model):

$$y_i = k_1 + k_2 \exp(-k_3 x_i)$$

在大 x_i 处 (即 $x_i \rightarrow \infty$),模型变为 $y_i \approx k_1$,而在 $x_i \rightarrow 0$ 时,模型简化为:

$$y_i = k_1 + k_2(1 - k_3 x_i)$$

或 $y_i = (k_1 + k_2) - k_2 k_3 x_i$

因此,用靠近 0 的 x_i 数据进行线性回归,可估计 k_1+k_2 和 $k_2 k_3$,这样,以上 k_1 、 k_2 和 k_3 的初步估值就全部确定。

③ 模型方程的变换——用多元线性回归确定参数初始值

先把非线性系统转变为线性系统,再进行线性最小二乘估计 (或多元线性回归),得到的线性结果作为非线性系统参数估计的初始值。这种线性化方法使用比较普遍。

④ 直接搜索法

如果对参数了解不多,可用像 LJ 优化技术的直接搜索法来为 Gauss-Newton 法提供很好的初始估计值。实际上,对于代数方程模型,直接搜索法用于确定最优参数估计很有效,但是如果需要估计参数中的不确定性,强烈推荐 Gauss-Newton 法。

(3) 数据平滑处理

在进行参数估计之前,应先对实验数据进行平滑处理,以便从数据移去测量误差(噪音)。尤其是微分法,为使计算导数时误差最小,强烈建议通过多项式拟合对数据平滑处理。平滑数据和多项式拟合的最好和最容易的方法是使用平滑三次样条函数。

7.3.2 动力学参数估计

这里的动力学参数估计主要包括化学反应动力学参数估计、生化反应动力学参数估计和医学工程药理动力学参数估计等。

7.3.2.1 化学反应动力学参数估计

反应速率数据分析 (analysis of rate data) 是化学工程中的一个重要内容,其任务就是进行反应动力学参数估计。

获取速率数据两种常用的反应器类型是:

(1) 间歇反应器:主要用于均相反应,通常在反应过程 (非稳态操作) 中,实验测量不同时间的浓度、压力和/或体积;

(2) 微分反应器:主要用于流-固反应,通常在微分反应器的稳态操作下进行实验,对不

同的进料条件测量产品的浓度。

反应速率数据分析一般有微分法 (the differential method)、积分法 (the integral method)、初始速率法 (the method of initial rates) 和半衰期法 (the method of half-lines) 等。有关动力学实验装置及实验方法可参阅反应工程方面的文献[6~11]。从数学的角度看, 反应速率数据分析 (参数估计) 主要有微分法和积分法两种 (见例 7-1)。

反应器模型方程的参数估计

有一组参数和时变参数的化学反应器模型方程, 可分为表 7-1 中的几类。

表 7-1 几种反应器模型方程

项 目	线性系统的例子	非线性系统的例子
代数方程	有一级动力学的稳态 CSTR 模型 代数解和优化 (最小二乘法)	有复杂动力学的稳态 CSTR 模型 数值解和优化 (最小二乘法或极大似然法)
微分方程	发生一级反应的间歇反应器模型 有分析参数优化的解析解或有数值参数优化 (最小二乘法或极大似然法)	发生复杂反应的间歇反应器模型 数值积分和参数优化 (最小二乘法或极大似然法)

活化能和指前因子的估计

一般先分析各种温度 T_i 下的数据, 得到反应速率常数 k_i , 接着由 Arrhenius 公式确定活化能和指前因子。如果直接将参数对温度的依赖关系代入速率方程, 并在所有温度下同时对全部数据进行回归, 就可避免采用上述分成两步的做法。但这样做会大大增加方程的非线性特性。为了减少计算时由于频率因子和活化能之间的强相关性而引起的困难, 往往进行如下变换 (Kittrell, 1970^[12])。

$$k_0 \exp\left(-\frac{E}{RT}\right) = k_0 \exp\left(-\frac{E}{RT}\right) \exp\left[-\frac{E}{R}\left(\frac{1}{T} - \frac{1}{\bar{T}}\right)\right] = k_0^* \exp\left[-\frac{E}{R}\left(\frac{1}{T} - \frac{1}{\bar{T}}\right)\right]$$

式中, \bar{T} 为平均温度。

【例 7-1】 分别用微分法和积分法重新求解[例 3-17]中的问题。

反应物 A 在一等温间歇反应器中发生反应: $A \rightarrow \text{产物}$, 测得反应器中不同时间下反应物 A 的浓度 C_A , 如表 7-2 所示。

表 7-2 等温间歇反应器中的浓度变化^[7]

时间 t/s	0	20	40	60	120	180	300
$C_A/(\text{mol/L})$	10	8	6	5	3	2	1

试根据表中数据确定其反应速率方程。

动力学模型

$$-r_A = kC_A^n \quad (\text{a})$$

程序说明 在[例 3-17]的程序中, 先用最小二乘样条拟合函数 `spap2()` 拟合得到 B 样条函数 `sp`, 再用 `fnder()` 计算 `sp` 的导函数 `pp`, 然后用 `fnval()` 计算 `pp` 在 t 处的导数值 (即反应速率) dC_A/dt 。最后, 用 `polyfit()` 进行单变量线性回归得到参数 k 和 n 。这是微分法结合线性回归的例子。

本例先用微分法结合非线性回归进行参数估计:

先用微分法求 dC_A/dt (同例 3-17), 再用非线性最小二乘法函数 `lsqnonlin()` 直接由非线性模型 (a) 估计得到动力学参数 k 和 n 及其参数置信区间。该程序为 `KineticsEst1_Diff.m`。

也可以用积分法结合非线性回归进行参数估计:

$$\text{由 (a) 式得} \quad \frac{dC_A}{dt} = -kC_A^n \quad (\text{b})$$

对于给定的参数 k 和 n , 可用 `ode45()` 积分得到对应于实验点 $t = [0 \ 20 \ 40 \ 60 \ 120 \ 180 \ 300]$ 处的浓度计算值。因此, 可以利用 `lsqnonlin()`, 以浓度测量值与计算值的残差平方和为优化目标, 搜索得到最优的动力学参数 k 和 n , 其中, 每搜索一步(迭代一次), 即调用一次 `ode45()` 得到浓度计算值。此程序为 `KineticsEst1_Int.m`。

程序清单 *KineticsEst1_Diff.m*

```
function KineticsEst1_Diff    % 用微分法进行反应速率分析得到速率常数 k 和反应级数 n
clear all; clc
t=[0 20 40 60 120 180 300]; CAm=[10 8 6 5 3 2 1];    % 动力学数据

% 用最小二乘样条拟合法计算微分 dCA/dt--使用不经过实验点的 B 样条插值函数
knots = 3; K = 3;    % 三次 B 样条
sp = spap2(knots, K, t, CAm);
pp = fnder(sp); dCAdt = fnval(pp, t);    % 计算 B 样条函数的导函数及其在 t 处的导函数值
rAm = dCAdt;

% 绘制浓度拟合曲线
ti = linspace(t(1), t(end), 200); CAi = fnval(sp, ti);
plot(t, CAm, 'ro', ti, CAi, 'b-'), xlabel('t'), ylabel('C_A'), legend('实验值', 'B 样条拟合')
beta0 = [0.0053 1.39];    % 参数初值
[beta, resnorm, rcsidual, exitflag, output, lambda, jacobian] = ...
    lsqnonlin(@OptObjFunc, beta0, [], [], [], rAm, CAm)    % 非线性拟合
ci = nlparci(beta, residual, jacobian)    % 参数的置信区间

% 显示参数辨识结果
fprintf('Estimated Parameters:\n'), fprintf('\tk = %.4f ± %.4f\n', beta(1), ci(1, 2) - beta(1))
fprintf('\tn = %.2f ± %.2f\n', beta(2), ci(2, 2) - beta(2))

% 绘制反应速率拟合曲线
figure, plot(t, rAm, 'ro', t, Rate(CAm, beta), 'b*')
xlabel('t'), ylabel('dC_Adt'), legend('Experiment', 'Kinetic Model')

% -----
function f = OptObjFunc(beta, rAm, CAm)
rAc = Rate(CAm, beta);
f = rAc - rAm;

% -----
function rA = Rate(CA, beta)
rA = -beta(1)*CA.^beta(2);    % -rA = -dCAdt = k*CA^n, 其中 k=beta(1), n=beta(2)
```

KineticsEst1_Int.m

```
function KineticsEst1_Int    % 用积分法进行反应速率分析得到速率常数 k 和反应级数 n
clear all; clc
global Cam
t=[0 20 40 60 120 180 300]; CA0=[10 8 6 5 3 2 1]; % 动力学数据
% 非线性拟合
beta0=[0.0053 1.39]; tspan=[0 20 40 60 120 180 300]; CA0=10;
[beta, resnorm, resid, exitflag, output, lambda, jacobian]=...
    lsqnonlin(@OptObjFunc, beta0, [], [], [], tspan, CA0, CA0)
ci = nlparci(beta, resid, jacobian)
% 拟合效果图(实验与拟合的比较)
[t4plot CA4plot]=ode45(@KineticsEqs, [tspan(1) tspan(end)], CA0, [], beta);
plot(t, CA0, 'bo', t4plot, CA4plot, 'k-')
legend('Exp', 'Model'), xlabel('时间 t, s'), ylabel('浓度 C_A, mol/L')
% 残差关于拟合值的残差图
[t CAC]=ode45(@KineticsEqs, tspan, CA0, [], beta);
figure, plot(CAC, resid, '*'), xlabel('浓度拟合值 (mol/L)'), ylabel('残差 R (mol/L)'), reline(0,0)
% 参数辨识结果
fprintf('Estimated Parameters:\n'), fprintf('\tk = %.4f ± %.4f\n', beta(1), ci(1, 2) - beta(1))
fprintf('\tn = %.2f ± %.2f\n', beta(2), ci(2, 2) - beta(2))
% -----
function f = OptObjFunc(beta, tspan, CA0, CA0)
[t CAC]=ode45(@KineticsEqs, tspan, CA0, [], beta);
f = CAC - CA0;
% -----
function dCAdt = KineticsEqs(t, CA, beta)
dCAdt = -beta(1)*CA^beta(2); % k = beta(1), n = beta(2)
```

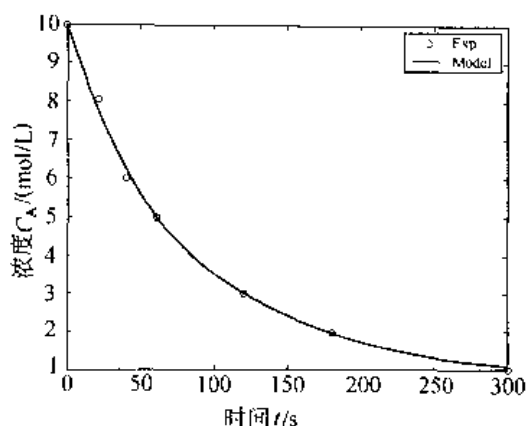
计算结果 辨识得到的动力学参数为:

微分法	$k = 0.0055 \pm 0.0025$	$n = 1.37 \pm 0.22$
积分法	$k = 0.0047 \pm 0.0016$	$n = 1.46 \pm 0.19$

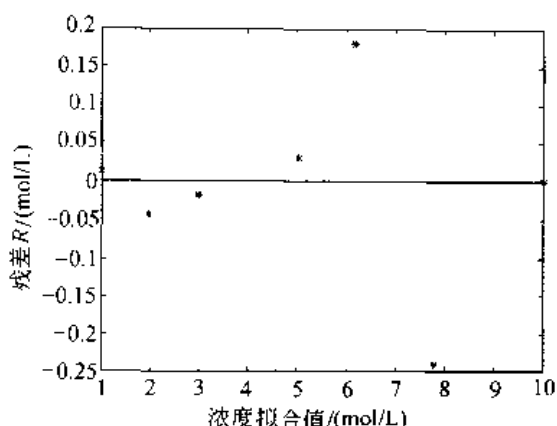
可见, 由积分法得到的参数置信区间要比微分法的窄, 从这点看, 积分法较佳。

用积分法 (KineticsEst1_Int.m) 得到的拟合效果如图 7-1 (微分法的拟合效果图从略)。从残差分析图 7-1(b)可以看出, 各次实验的残差均匀地分布在零线的上下侧, 比较满意。

与[例 3-17]的解法不同, 这里使用非线性最小二乘法直接由非线性模型估计其参数, 并计算出参数的置信区间, 实际上是最优化方法和统计学的综合运用。



(a) 拟合曲线



(b) 残差分析

图 7-1 最小二乘拟合法（积分法）

【例 7-2】 乙烯（E）加氢（H）制乙烷（EA）反应如下： $\text{H}_2 + \text{C}_2\text{H}_4 \rightarrow \text{C}_2\text{H}_6$ ，在微分反应器中测得的实验数据如表 7-3 所示。

表 7-3 乙烯加氢动力学实验数据^[9]

实验号	反应速率 $-r_A$ mol/(kg 催化剂·s)	p_1 /atm	p_{FA} /atm	p_H /atm
1	1.0400	1.0000	1.0000	1.0000
2	3.1300	1.0000	1.0000	3.0000
3	5.2100	1.0000	1.0000	5.0000
4	3.8200	3.0000	1.0000	3.0000
5	4.1900	5.0000	1.0000	3.0000
6	2.3910	0.5000	1.0000	3.0000
7	3.8670	0.5000	0.5000	5.0000
8	2.1990	0.5000	3.0000	3.0000
9	0.7500	0.5000	5.0000	1.0000

注：1atm=101.3kPa。

反应速率方程为：

$$-r_A = \frac{k p_E p_H}{1 + K_E p_E} \quad (\text{a})$$

试根据表中数据确定动力学参数 k 和 K_E 。

程序说明 先进行线性回归：取（a）式的倒数，并把 p_E 、 p_H 移向左侧，则

$$R_A = b_1 + b_2 p_E \quad (\text{b})$$

其中

$$R_A = \frac{p_E p_H}{-r_A}, \quad b_1 = \frac{1}{k}, \quad b_2 = \frac{K_E}{k} \quad (\text{c})$$

在 b_1 、 b_2 中含有未知参数 k 和 K_E ，所以它们成为新的未知量。按表 7-3 中的数据，用线性最小二乘法估计（b）式的 b_1 、 b_2 ，再由（c）式求出 k 和 K_E 。

然后，由上述线性回归得到的 k 和 K_E 作为非线性回归的初值，利用非线性最小二乘法的函数 `lsqnonlin()` 直接估计非线性模型（a）中的参数 k 和 K_E ，并计算其置信区间。

程序清单 *KineticsEst2.m*

```
function KineticsEst2
```

```

clear all; clc

% 实验数据
rA = [1.04 3.13 5.21 3.82 4.19 2.391 3.867 2.199 0.75]; Pe = [1 1 1 3 5 0.5 0.5 0.5 0.5]; Ph = [1 3
5 3 3 3 5 3 1];

% 线性拟合
RA = Pe.*Ph./rA; y = RA'; X = [ones(size(y)) Pe'];
b = X\y; k = 1/b(1); KE = b(2)*k;

% 非线性拟合
beta0 = [k KE]
[beta, resnorm, residual, exitflag, output, lambda, jacobian] = lsqnonlin(@ObjFunc, beta0, [], [], [],
rA, Pe, Ph)
ci = nlparci(beta, residual, jacobian)

% 参数辨识结果
fprintf('Estimated Parameters:\n'), fprintf('\tk = %.3f ± %.3f\n', beta(1), ci(1, 2) - beta(1))
fprintf('\tKE = %.3f ± %.3f\n', beta(2), ci(2, 2) - beta(2))
fprintf('\tThe sum of the squares is: %.3f, resnorm)

% -----
function f = ObjFunc(beta, rA, Pe, Ph)
f = rA - beta(1)*Pe.*Ph./(1+beta(2)*Pe);

```

计算结果 $k = 3.187 \pm 0.288$, $K_E = 2.101 \pm 0.264$; 残差平方和: 0.042。拟合残差见图 7-2 所示, 由图可见, 拟合效果比较满意。

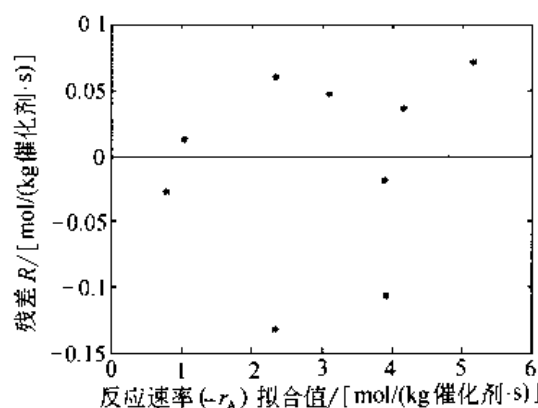


图 7-2 拟合残差

【例 7-3】用初始速率法进行动力学分析^[13]

对于反应系统 $A + 2B \rightarrow C$, 其动力学模型如 (1) 式, 从间歇反应器中获得的初始反应速率数据如表 7-4 所示, 试拟合该动力学模型参数 k , n 和 m 。

$$-r_A = kp_A^n p_B^m \quad (1)$$

表 7-4 初始压力和初始反应速率

实验序号	p_{A0}/torr	p_{B0}/torr	$-r_{A0}/\text{torr s}^{-1}$	实验序号	p_{A0}/torr	p_{B0}/torr	$-r_{A0}/\text{torr s}^{-1}$
1	6	20	0.420	6	10	10	0.639
2	8	20	0.647	7	10	20	0.895
3	10	20	0.895	8	10	40	1.265
4	12	20	1.188	9	10	60	1.550
5	16	20	1.811	10	10	100	2.021

① 1 torr=133.322 Pa。

程序说明 用非线性回归拟合参数：先对(1)式两边取对数进行线性化，再用多元线性回归函数 regress()估计参数 k 、 n 和 m ，然后以此作为初值用 lsqnonlin()进行非线性回归。其中 ObjFunc()定义非线性回归最优化目标函数，供 lsqnonlin()调用，而 Rate()定义速率方程(1)。

程序清单 见光盘中的 KineticsEst3.m。

计算结果 拟合残差图如图 7-3，拟合参数值(95%置信度)为： $k = 0.0064 \pm 0.0004$ ， $n = 1.49 \pm 0.02$ ， $m = 0.50 \pm 0.01$ 。残差平方和为 $3.0\text{e-}004$ 。

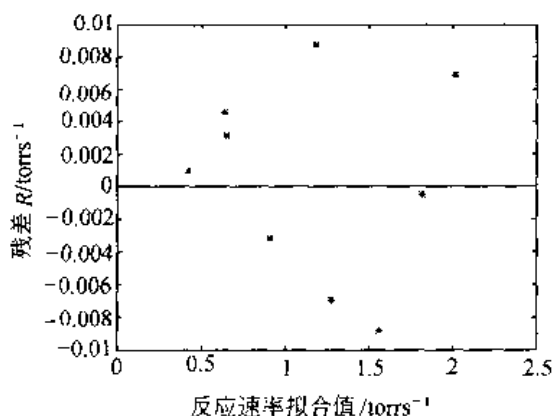


图 7-3 拟合残差图

【例 7-4】一氧化氮催化还原^[3]

在微分流动反应器中进行一氧化氮催化还原反应： $\text{NO} + \text{H}_2 \rightleftharpoons \text{H}_2\text{O} + \frac{1}{2}\text{N}_2$ ，以氮气作为稀释剂，反应压力为常压，实验在 375℃、400℃ 和 425℃ 的不同温度下进行，测定在不同氢气分压 p_{H_2} 和一氧化氮分压 p_{NO} 下的反应速率和 NO 总转化率，如表 7-5 所示。一个被吸附的一氧化氮分子与相邻的一个被吸附的氢气分子之间的反应可用 Langmuir-Hinshelwood 反应速率模型表示：

$$r = \frac{kK_{\text{H}_2}K_{\text{NO}}p_{\text{H}_2}p_{\text{NO}}}{(1 + K_{\text{NO}}p_{\text{NO}} + K_{\text{H}_2}p_{\text{H}_2})^2} \quad (\text{a})$$

式中， r 为反应速率； $\text{mol}/(\text{min} \cdot \text{g 催化剂})$ ； p_{H} 为氢气分压 atm； p_{NO} 为一氧化氮分压，atm； k 为 $k_0 \exp(-\frac{E_1}{RT})$ 为表面反应的正向反应速率常数， $\text{mol}/(\text{min} \cdot \text{g 催化剂})$ ； K_{H_2} 为 $k_0 \exp(-\frac{E_3}{RT})$

为氢气的吸附平衡常数, atm^{-1} ; K_{NO} 为 $k_{02} \exp(-\frac{E_2}{RT})$ 为氮气的吸附平衡常数; atm^{-1} ; E_1 , E_2 和 E_3 单位为 cal/mol ; R 为理想气体常数 ($8.314 \text{ J}/(\text{mol}\cdot\text{K})$ 或 $1.987 \text{ cal}/(\text{mol}\cdot\text{K})$).

表 7-5 一氧化氮催化还原的实验数据 ($T=375\text{ }^{\circ}\text{C}$, 催化剂量 = 2.39 g)

$p_{\text{H}_2}/\text{atm}$	p_{NO}/atm	反应速率 $r \times 10^5 \text{ mol}/(\text{min}\cdot\text{g 催化剂})$	NO 总转化率/%
0.0092	0.0500	1.6000	1.96
0.0136	0.0500	2.5600	2.36
0.0197	0.0500	3.2700	2.99
0.0280	0.0500	3.6400	3.54
0.0291	0.0500	3.4800	3.41
0.0389	0.0500	4.4600	4.23
0.0485	0.0500	4.7500	4.78
0.0500	0.0092	1.4700	14.0
0.0500	0.0184	2.4800	9.15
0.0500	0.0298	3.4500	6.24
0.0500	0.0378	4.0600	5.40
0.0500	0.0491	4.7500	4.30

注：表 7-4 数据来源：Ayen 和 Peters (1962)：Ayen, R. J. and Peters M. S., “Catalytic Reduction of Nitric Oxide”, Ind Eng Chem Proc Des Dev, 1, 204~207 (1962).

程序说明

先进行线性回归：

取 (a) 式的倒数，并把 $p_{\text{H}_2}p_{\text{NO}}$ 移向左侧，然后使两边都变成 1/2 次方，则：

$$R = b_1 + b_2p_{\text{H}_2} + b_3p_{\text{NO}} \tag{b}$$

$$\text{式中, } R = \sqrt{\frac{p_{\text{H}_2}p_{\text{NO}}}{r}}, \quad b_1 = \frac{1}{\sqrt{kK_{\text{H}_2}K_{\text{NO}}}}, \quad b_2 = \frac{K_{\text{H}_2}}{\sqrt{kK_{\text{H}_2}K_{\text{NO}}}}, \quad b_3 = \frac{K_{\text{NO}}}{\sqrt{kK_{\text{H}_2}K_{\text{NO}}}} \tag{c}$$

在 b_1 、 b_2 、 b_3 中含有未知参数 k 、 K_{H_2} 和 K_{NO} ，所以它们成为新的未知量。按表 7-4 中的数据，用线性最小二乘法估计 (b) 式的 b_1 、 b_2 、 b_3 ，由 (c) 再求出 k 、 K_{H_2} 和 K_{NO} 。

然后，由上述线性回归得到的 k 、 K_{H_2} 和 K_{NO} 作为非线性回归的初值，利用非线性最小二乘函数 lsqnonlin() 直接估计非线性模型 (a) 中的参数 k 、 K_{H_2} 和 K_{NO} ，并计算其置信区间。

本例从一个等温 ($T = 375\text{ }^{\circ}\text{C}$) 下的实验数据估计参数 k ， K_{H_2} 和 K_{NO} 。如果再测量其他温度（如 390°C 、 420°C ）下的实验数据，并估计对应温度下的参数 k ， K_{H_2} 和 K_{NO} ，则从所有这些数据 T_i 、 k_i ， $K_{\text{H}_2,i}$ 和 $K_{\text{NO},i}$ 可直接估计参数 k_{01} 、 k_{02} 、 k_{03} 、 E_1 、 E_2 和 E_3 。

程序清单 *KineticsEst4.m*

```
function KineticsEst4
clear all; clc
load ChemKineticsData      % Load experimental data
PH2 = ExpData(:, 1);  PNO = ExpData(:, 2);  r = ExpData(:, 3);

% 用多变量线性回归方法估计动力学参数
```

```

R = sqrt(PH2.*PNO./r); y = R; X = [ones(size(y)) PH2 PNO];
[b, bint] = regress(y, X, 0.05); % 或 b = X\y
denom = 1/b(1); KH2 = b(2)*denom; KNO = b(3)*denom; k = denom^2/(KH2*KNO);

% 用 lsqnonlin()--求解非线性最小二乘法(非线性数据拟合)问题
lb = [0 0 0]; ub = [+inf +inf +inf]; beta0 = [k KH2 KNO];
[beta, resnorm, resid, exitflag, output, lambda, jacobian] = lsqnonlin(@ObjFun, beta0, lb, ub, [],
PH2, PNO, r);
ci = nlparci(beta, resid, jacobian);

% 模型适定性判别
Ne = length(r); Np = length(beta); [rho2, F] = rho2_F(k, r, resnorm, Ne, Np);

% 残差关于拟合值的残差图
rc = RateEqs(beta, PH2, PNO); plot(rc, resid, '*')
xlabel('反应速率拟合值, mol/(min g 催化剂)'), ylabel('残差 R, mol/(min g 催化剂)'), reline(0, 0)

% 参数辨识结果
fprintf('\n\nEstimated Parameters:\n')
fprintf('\tk = %.4f ± %.4f\n', beta(1), ci(1, 2) - beta(1))
fprintf('\tKH2 = %.2f ± %.2f\n', beta(2), ci(2, 2) - beta(2))
fprintf('\tKNO = %.2f ± %.2f\n', beta(3), ci(3, 2) - beta(3))

% -----
function f = ObjFun(beta, PH2, PNO, r)
rc = RateEqs(beta, PH2, PNO); f = r - rc;

% -----
function rc = RateEqs(beta, PH2, PNO) % Rate equation
rc = beta(1)*beta(2)*beta(3)*PH2.*PNO./(1+beta(3)*PNO+beta(2)*PH2).^2;

```

计算结果 参数拟合值(95%置信度)为 $k = 0.0005 \pm 0.0002$, $K_{H_2} = 19.03 \pm 8.23$, $K_{NO} = 14.33 \pm 8.19$ 。残差图示于图 7-4 中。可见, 置信区间比较宽、残差 R 比较大。

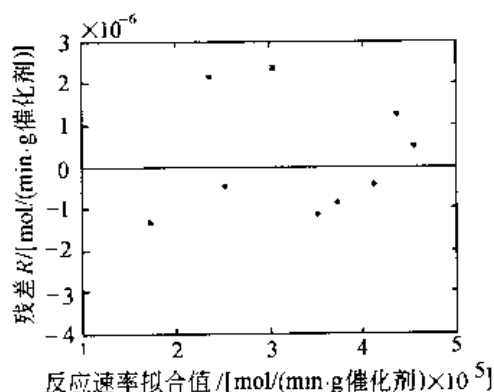


图 7-4 残差

【例 7-5】 在一个等温间歇反应器中进行复杂反应，描述该反应系统的七个微分方程共有 5 个参数 k_1, k_2, k_3, k_4, k_5 ，初始状态为 $x(0) = [0.1883 \ 0.2507 \ 0.0467 \ 0.0899 \ 0.1804 \ 0.1394 \ 0.1046]^T$ ，试根据表 7-6 的数据，利用最优化方法估计出 k 参数^[14]。

表 7-6 各组成的动态数据

t/h	x_1	x_2	x_3	x_4	x_5	x_6
0	0.1883	0.2507	0.0467	0.0899	0.1804	0.1394
0.0100	0.2047	0.2382	0.0866	0.0866	0.1729	0.1297
0.0200	0.2181	0.2382	0.0856	0.0856	0.1680	0.1205
0.0300	0.2291	0.2382	0.0863	0.0863	0.1647	0.1123
0.0400	0.2382	0.2382	0.0878	0.0878	0.1623	0.1053
0.0500	0.2459	0.2382	0.0899	0.0899	0.1604	0.0995
0.0600	0.2523	0.2382	0.0921	0.0921	0.1588	0.0948
0.0700	0.2576	0.2382	0.0945	0.0945	0.1574	0.0911
0.0800	0.2622	0.2382	0.0968	0.0968	0.1561	0.0882
0.0900	0.2660	0.2382	0.0989	0.0989	0.1548	0.0859
0.1000	0.2692	0.2382	0.1010	0.1010	0.1537	0.0842
0.1100	0.2719	0.2382	0.1028	0.1028	0.1525	0.0830
0.1200	0.2742	0.2382	0.1045	0.1045	0.1515	0.0821
0.1300	0.2761	0.2382	0.1060	0.1060	0.1505	0.0814
0.1400	0.2777	0.2382	0.1074	0.1074	0.1495	0.0810
0.1500	0.2790	0.2382	0.1086	0.1086	0.1487	0.0807
0.1600	0.2801	0.2382	0.1096	0.1096	0.1479	0.0805
0.1700	0.2811	0.2382	0.1106	0.1106	0.1471	0.0803
0.1800	0.2819	0.2382	0.1114	0.1114	0.1465	0.0803
0.1900	0.2825	0.2382	0.1121	0.1121	0.1458	0.0803
0.2000	0.2830	0.2382	0.1127	0.1127	0.1453	0.0803

数学模型

$$\frac{dx_1}{dt} = k_5 - qx_1 - k_1x_1x_2 - k_4x_1x_6\sqrt{0.9}$$

$$\frac{dx_2}{dt} = 7.0 - qx_2 - k_1x_1x_2 - 2k_2x_2x_3$$

$$\frac{dx_3}{dt} = 1.75 - qx_3 - k_2x_2x_3$$

$$\frac{dx_4}{dt} = -qx_4 + 2k_1x_1x_2 - k_3x_4x_5$$

$$\frac{dx_5}{dt} = -qx_5 + 3k_2x_2x_3 - k_3x_4x_5$$

$$\frac{dx_6}{dt} = -qx_6 + 2k_3x_4x_5 - k_4x_1x_6\sqrt{0.9}$$

$$\frac{dx_7}{dt} = -qx_7 + 2k_4x_1x_6\sqrt{0.9}$$

式中， $q = 8.75 + k_5$ 。

程序说明 先以 $k = [5.0 \ 5.0 \ 5.0 \ 5.0 \ 5.0]^T$ 为初值，分别使用 `fmincon()` 和 `lsqnonlin()` 进行

参数估计，然后以 `fmincon()` 的结果为初值，使用 `lsqnonlin()` 进行参数估计。其中 `ObjFunc4Fmincon()` 和 `ObjFunc4LNL()` 分别供 `fmincon()` 和 `lsqnonlin()` 调用，而 `Output.m` 用于输出 `lsqnonlin()` 的结果。 $y(1) = x_1$, $y(2) = x_4$, $y(3) = x_5$, $y(4) = x_6$ ，相应地，`yexp(1~4)` 表示 x_1 、 x_4 、 x_5 、 x_6 的实验数据。

程序清单 *KineticsEst5.m* (表 7-6 中的数据存于文件 *KineticsData1.m* 中)

```
function KineticsEst5
clear all; clc
k0 = [0.5 0.5 0.5 0.5 0.5]; % 参数初值
lb = [0 0 0 0 0]; ub = [+inf +inf +inf +inf +inf]; % lb、ub: 参数下限和上限
x0 = [0.1883 0.2507 0.0467 0.0899 0.1804 0.1394 0.1046];
KineticsData1; yexp = ExpData(:, 2:5); % yexp: 实验数据[x1 x4 x5 x6]

% 使用函数 fmincon() 进行参数估计
[k, fval, flag] = fmincon(@ObjFunc4Fmincon, k0, [], [], [], [], lb, ub, [], [], x0, yexp);
fprintf('\n 使用函数 fmincon() 估计得到的参数值为:\n')
fprintf('\tk1 = %.4f\n', k(1)), fprintf('\tk2 = %.4f\n', k(2)), fprintf('\tk3 = %.4f\n', k(3))
fprintf('\tk4 = %.4f\n', k(4)), fprintf('\tk5 = %.4f\n', k(5))
fprintf(' The sum of the squares is: %.1e\n\n', fval)
k_fmincon = k;

% 使用函数 lsqnonlin() 进行参数估计
[k, resnorm, residual, exitflag, output, lambda, jacobian] = ...
    lsqnonlin(@ObjFunc4LNL, k0, lb, ub, [], x0, yexp);
ci = nlparci(k, residual, jacobian);
fprintf('\n\n 使用函数 lsqnonlin() 估计得到的参数值为:\n'), Output

% 以函数 fmincon() 估计得到的结果为初值，使用函数 lsqnonlin() 进行参数估计
k0 = k_fmincon;
[k, resnorm, residual, exitflag, output, lambda, jacobian] = lsqnonlin(@ObjFunc4LNL, k0, lb, ub,
[], x0, yexp);
ci = nlparci(k, residual, jacobian);
fprintf('\n\n 以 fmincon() 的结果为初值，使用函数 lsqnonlin() 估计得到的参数值为:\n')
Output

% -----
function f = ObjFunc4Fmincon(k, x0, yexp)
tspan = [0.00 : 0.01 : 0.20];
[t x] = ode45(@KineticEqs, tspan, x0, [], k);
y(:, 1) = x(:, 1); y(:, 2:4) = x(:, 4:6);
f = sum((y(:, 1) - yexp(:, 1)).^2) + sum((y(:, 2) - yexp(:, 2)).^2) ...
    + sum((y(:, 3) - yexp(:, 3)).^2) + sum((y(:, 4) - yexp(:, 4)).^2);
```

```

% -----
function f = ObjFunc4LNL(k, x0, yexp)
tspan = [0.00 : 0.01 : 0.20];
[t x] = ode45(@KineticEqs, tspan, x0, [], k);
y(:, 1) = x(:, 1); y(:, 2:4) = x(:, 4:6);
f1 = y(:, 1) - yexp(:, 1); f2 = y(:, 2) - yexp(:, 2);
f3 = y(:, 3) - yexp(:, 3); f4 = y(:, 4) - yexp(:, 4);
f = [f1; f2; f3; f4];

% -----
function dxdt = KineticEqs(t, x, k)
q = 8.75 + k(5);
dxdt = ...
[ ( k(5) - q*x(1) - k(1)*x(1)*x(2) - k(4)*x(1)*x(6)*sqrt(0.9) )
  ( 7.0 - q*x(2) - k(1)*x(1)*x(2) - 2*k(2)*x(2)*x(3) )
  ( 1.75 - q*x(3) - k(2)*x(2)*x(3) )
  ( -q*x(4) + 2*k(1)*x(1)*x(2) - k(3)*x(4)*x(5) )
  ( -q*x(5) + 3*k(2)*x(2)*x(3) - k(3)*x(4)*x(5) )
  ( -q*x(6) + 2*k(3)*x(4)*x(5) - k(4)*x(1)*x(6)*sqrt(0.9) )
  ( -q*x(7) + 2*k(4)*x(1)*x(6)*sqrt(0.9) )
];

```

计算结果

用 `fmincon()` 得到的参数值为 $k = [17.5179 \quad 72.2765 \quad 50.8686 \quad 22.6293 \quad 5.9822]^T$, 目标函数值为 $1.0e-006$ 。而用 `lsqnonlin()` 得到的参数值为 $k_1 = 16.4690 \pm 0.2555$, $k_2 = 65.8229 \pm 1.5618$, $k_3 = 47.0197 \pm 0.9899$, $k_4 = 17.9220 \pm 1.1701$, $k_5 = 5.7910 \pm 0.0480$ 。目标函数值为 $2.8e-005$ 。

可见, 使用 `fmincon()` 时, 收敛于较低的目标函数值, 说明在本例的情况下, `fmincon()` 的估计结果明显优于 `lsqnonlin()`。但是 `fmincon()` 并不计算参数的置信区间, 即缺乏统计信息。而函数 `lsqnonlin()` 虽然可计算出参数的置信区间, 但其参数估值不如 `fmincon()` 好。因此, 结合两种算法较佳, 即先用 `fmincon()` 计算最优参数值, 再把此参数值作为 `lsqnonlin()` 的初值进行参数估计, 则可得到最优参数值及其置信区间。这种方法并非在任何场合都有效, 但在有些情况确实可以改善估计结果, 因此在估计结果不理想时可以尝试这种方法。

以 `fmincon()` 的结果作为参数初值, 用 `lsqnonlin()` 估计得到的参数值为 $k_1 = 17.4900 \pm 0.0303$, $k_2 = 72.2996 \pm 0.1941$, $k_3 = 50.9086 \pm 0.1119$, $k_4 = 22.5339 \pm 0.1325$, $k_5 = 5.9799 \pm 0.0054$ 。对应的目标函数值为: $3.6e-007$ 。

【例 7-6】 甲苯催化加氢反应^[15] $A \xrightleftharpoons[k_2]{k_1} B \xrightarrow{k_3} C$, 请利用表 7-7 的浓度测量值, 辨识(估计)出参数 k_1 、 k_2 、 k_3 、 k_4 和 k_5 。已知初始浓度 $C_{A0} = 1$, $C_{B0} = 0$, $C_{C0} = 0$ (以纯甲苯开始反应)。

表中, C_A 、 C_B 和 C_C 分别是 A、B 和 C 的浓度。

表 7-7 甲苯催化实验数据

t/min	C_A	C_B	C_C	t/min	C_A	C_B	C_C
15	0.695	0.312	0.001	180	0.056	0.362	0.580
30	0.492	0.430	0.080	240	0.041	0.211	0.747
45	0.276	0.575	0.151	320	0.031	0.146	0.822
60	0.225	0.570	0.195	360	0.022	0.080	0.898
75	0.163	0.575	0.224	380	0.021	0.070	0.909
90	0.134	0.533	0.330	400	0.019	0.073	0.908
120	0.064	0.462	0.471				

数学模型

系统由三个微分方程描述:

$$\frac{dC_A}{dt} = -r_1 + r_2 \quad (\text{a})$$

$$\frac{dC_B}{dt} = r_1 - r_2 - r_3 \quad (\text{b})$$

$$\frac{dC_C}{dt} = r_3 \quad (\text{c})$$

其中, 反应速率为

$$r_1 = k_1 \theta_A, \quad r_2 = k_2 \theta_B, \quad r_3 = k_3 \theta_B \quad (\text{d})$$

表面覆盖率:

$$\theta_A = \frac{K_A C_A}{K_A C_A + C_B + K_C C_C}, \quad \theta_B = \frac{C_B}{K_A C_A + C_B + K_C C_C} \quad (\text{e})$$

目标函数

$$\min f = \sum_{i=1}^N \left[x(t_i) - \hat{x}(t_i) \right]^T \left[x(t_i) - \hat{x}(t_i) \right] \quad (\text{f})$$

程序说明 取初始向量 $\mathbf{K} = [0.02 \quad 0.005 \quad 0.025 \quad 1.5 \quad 1.25]^T$

程序清单 见光盘中的 *KineticsEst6.m*。

计算结果 $k_1 = 0.0238 \pm 0.0033$, $k_2 = 0.0042 \pm 0.0036$, $k_3 = 0.0098 \pm 0.0030$, $k_4 = 1.5773 \pm 0.8554$, $k_5 = 1.4566 \pm 0.6346$ 。残差平方和: $1.6\text{e-}002$ 。

7.3.2.2 生化反应动力学参数估计

【例 7-7】酶催化反应动力学^[3]

著名的 Michaelis-Menten 酶催化动力学方程为:

$$r = \frac{r_{\max} S}{K_m + S} \quad (1)$$

式中, r_{\max} 是最大反应速率; K_m 是 Michaelis 常数; S 是底物的浓度。

为了确定参数 r_{\max} 和 K_m , 通常在不同底物浓度下测定初始速率。在 30°C 和 $\text{pH} = 7.5$ 下用胰岛素对 BTEE (Benzoyl-L-Tyrosine Ethyl Ester) 进行水解反应, 测得的实验数据如下表。

试利用表 7-8 数据估计参数 r_{\max} 和 K_m 。

表 7-8 酶催化反应动力学实验数据 (数据来源: Blanch and Clark, 1996)

$S/(\mu\text{mol/L})$	20	15	10	5.0	2.5
$r/[\mu\text{mol}/(\text{L} \cdot \text{min})]$	330	300	260	220	110

程序说明 式 (1) 是一个非线性模型, 传统的方法是将该非线性模型变换为线性模型, 然后进行线性回归。但严格的方法采用非线性回归, 而线性回归的结果可作为非线性回归的

初值。对方程 (1) 进行线性化时, 可得到:

$$\left(\frac{1}{r}\right) = \frac{1}{r_{\max}} + \frac{K_m}{r_{\max}} \left(\frac{1}{S}\right) \quad (2)$$

程序先按方程 (2), 用线性最小二乘法估计得到参数 r_{\max} 和 K_m 。然后, 用非线性最小二乘法直接对方程 (1) 进行参数估计。线性回归的拟合程度可用相关系数评价。拟合曲线及残差见图 7-5 (a) 和图 7-5 (b)

程序清单 *EnzymeKinetEst.m*

```
function EnzymeKinetEst
clear all; clc

% 实验数据
S = [20 15 10 5 2.5]; rate = [330 300 260 220 110];

% 线性回归
y = 1./rate; X = [ones(size(y)) 1./S]; [b, bint, r, rint, stats] = regress(y, X);
rmax_LinReg = 1/b(1); Km_LinReg = rmax_LinReg * b(2);
R2_LinReg = stats(1); % R2_LinReg: 相关系数的平方值

% 非线性回归
x = S; y = rate;
b0 = [rmax_LinReg Km_LinReg]; % 以线性回归的结果作为非线性回归的初值
[beta, resid, J] = nlinfit(x, y, @myfun, b0);
ratec = myfun(beta, S); % 速率拟合值(计算值)

% 拟合效果图(实验与拟合的比较)
S4plot = linspace(S(1), S(end), 300); rate4plot = myfun(beta, S4plot);
plot(x, y, 'o', S4plot, rate4plot, '-'), legend('Exp', 'Model')
xlabel('S (μM)'), ylabel('r (μM/min)')

% 残差关于拟合值的残差图
figure, plot(ratec, resid, '*'), xlabel('拟合 (μM/min)'), ylabel('残差 R (μM/min)'), reline(0, 0)

% 参数辨识结果
fprintf('Estimated Parameters by Linear Regression:\n')
fprintf('\trmax = %.1f\n', rmax_LinReg), fprintf('\tKm = %.3f\n', Km_LinReg)
fprintf('\tR2 = %.3f\n', R2_LinReg), fprintf('Estimated Parameters by Nonlinear Regression:\n')
fprintf('\trmax = %.1f\n', beta(1)), fprintf('\tKm = %.3f\n', beta(2))

% -----
function yhat = myfun(beta, x)
yhat = beta(1)*x./(beta(2)+x); % beta(1)=rmax, beta(2)=Km
```

计算结果 线性回归: $r_{\max} = 496.3$, $K_m = 8.368$, $R^2 = 0.963$ (相关系数的平方值);
非线性回归: $r_{\max} = 420.2$, $K_m = 5.705$ 。

对非线性模型（1）进行线性化时，也可以得到以下与方程（2）不同的形式：

$$r = r_{\max} - K_m \left(\frac{r}{S} \right) \quad (3)$$

$$\left(\frac{S}{r} \right) = \frac{K_m}{r_{\max}} + \frac{1}{r_{\max}} S \quad (4)$$

尽管方程（2）、（3）和（4）都是由非线性模型（1）变换而来，但是，用线性最小二乘法对这些方程估计得到的结果是不同的（对方程（3）和（4）的估计，请读者自己完成）。

对于非线性模型，用线性回归方法只能得到近似的结果，因而常作为非线性回归的初值。

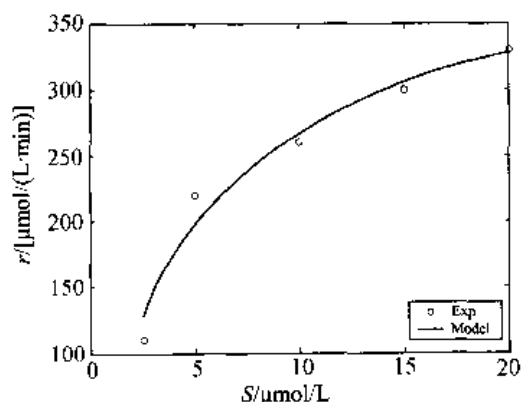


图 7-5(a) 非线性拟合曲线

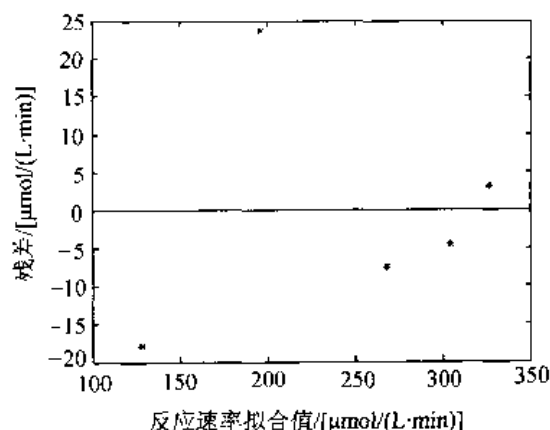


图 7-5(b) 残差

【例 7-8】生化反应动力学参数估计^[16]

在一连续发酵罐中固定进料（消毒）浓度、pH、空气流量（aeration）和温度等参数不变，调节稀释速度。当限制底物浓度为 1200 mg/L，发酵罐工作容积为 9.8 L 时，测得的数据如表 7-9。请估计下列改进 Monod 方程中的动力学常数 K_s 、 μ_m 和 k_d 及产率系数 Y ：

$$\mu = \frac{\mu_m S}{K_s + S} - k_d$$

表 7-9 连续发酵罐实验数据

进料流量 F (L/h)	出口底物浓度 S (mg/L)	干细胞密度 X (mg/L)
0.79	36.9	487
1.03	49.1	490
1.31	64.4	489
1.78	93.4	482
2.39	138.8	466
2.68	164.2	465

数学模型

对底物(substrate)和生物量 (biomass) 分别列出质量平衡方程：

$$V \frac{dX}{dt} = FX_0 - FX + V \frac{\mu_m SX}{K_s + S} - k_d XV \quad (1)$$

$$V \frac{dS}{dt} = FS_0 - FS - V \frac{\mu_m SX}{Y(K_s + S)} \quad (2)$$

式中, F = 进料流量=出口流量 = F_0 ; S_0 和 X_0 分别为进料中底物浓度和干细胞密度; X 和 S 分别为发酵罐出口处底物浓度和干细胞密度。

稳态时, $dX/dt=0$, $dS/dt=0$, 则由上面两个方程得

$$D(X - X_0) = \frac{\mu_m SX}{K_s + S} - k_d X \quad (3)$$

$$DY(S_0 - S) = \frac{\mu_m SX}{K_s + S} \quad (4)$$

由(4)变形得

$$\frac{X}{D(S_0 - S)} = \frac{Y}{\mu_m} + \frac{K_s Y}{\mu_m S} \quad (5)$$

把(4)代入(3), 并考虑 $X_0=0$ (原料已消毒) 得

$$DX = DY(S_0 - S) - k_d X$$

即

$$\frac{S_0 - S}{X} = -\frac{1}{Y} + \frac{k_d}{Y D} \quad (6)$$

根据方程(5)作 $\frac{X}{D(S_0 - S)} \sim \frac{1}{S}$ 的线性回归, 得到 $\frac{Y}{\mu_m} = a(1)$ 和 $\frac{K_s Y}{\mu_m} = a(2)$, 其中 $a(1)$ 和 $a(2)$

为回归系数。

根据方程(6)作 $\frac{S_0 - S}{X} \sim \frac{1}{Y}$ 的线性回归 $\frac{1}{Y} = b(1)$ 和 $\frac{k_d}{Y} = b(2)$, 其中 $b(1)$ 和 $b(2)$ 为回归系数。

则 $Y = \frac{1}{b(1)}$, $k_d = Y * b(2)$, $\mu_m = \frac{Y}{a(1)}$, $K_s = \frac{a(2)}{a(1)}$ (7)

程序说明 用多元线性回归函数 regress() 分别对方程(5)和(6)进行线性回归, 再根据(7)式得到动力学参数 K_s 、 μ_m 、 k_d 和 Y 。然后以此作为非线性回归的初值, 调用 lsqnonlin() 进行非线性回归, 其中 ObjFun() 定义其目标函数。在执行优化过程中, 每迭代(搜索)一次, 即调用一次 fsolve() 求解方程组(3)和(4)得出 S 和 X 的模拟值。

程序清单 见光盘中的 BioKineticsEst.m。

计算结果 用 regress() 估计得: $Y = 0.455$, $k_d = 0.007 \text{ h}^{-1}$, $\mu_m = 0.77 \text{ h}^{-1}$, $K_s = 289 \text{ mg/L}$, 以此作为初值, 由 lsqnonlin() 进行非线性参数估计得: $Y = 0.455 \pm 0.006$, $k_d = 0.007 \pm 0.002 \text{ h}^{-1}$, $\mu_m = 0.78 \pm 0.14 \text{ h}^{-1}$, $K_s = 289 \pm 77 \text{ mg/L}$ 。其残差平方和为 $5.2\text{e}+001$ 。

【例 7-9】青霉素发酵过程动力学 (fermentation kinetics) [17]

在间歇发酵罐中研究青霉素发酵动力学, 两组实验数据如表 7-10 所示。

表 7-10 青霉素发酵实验数据

t/h	第 1 组实验		第 2 组实验	
	$Y_1 \text{ exp}$	$Y_2 \text{ exp}$	$Y_1 \text{ exp}$	$Y_2 \text{ exp}$
0	0.40	0	0.18	0
10		0	0.12	0
22	0.99	0.0089	0.48	0.0089
34		0.0732	1.46	0.0642
46	1.95	0.1446	1.56	0.2266
58		0.5230	1.73	0.4373
70	2.52	0.6854	1.99	0.6943
82		1.2566	2.62	1.2459

续表

t/h	第 1 组实验		第 2 组实验	
	$y_1 \text{ exp}$	$y_2 \text{ exp}$	$y_1 \text{ exp}$	$y_2 \text{ exp}$
94	3.09	1.6118	2.88	1.4315
106		1.8243	3.43	2.0402
118	4.06	2.2170	3.37	1.9278
130		2.2758	3.92	2.1848
142	4.48	2.8096	3.96	2.4204
154		2.6846	3.58	2.4615
166	4.25	2.8738	3.58	2.2830
178		2.8345	3.34	2.7078
190	4.36	2.8828	3.47	2.6542

数学模型 在间歇反应器中微生物 *Penicillium chrysogenum* 在仔细控制条件下生长。在发酵过程中, 细胞生长速率符合逻辑定律 (logistic law)

$$\frac{dy_1}{dt} = k_1 y_1 \left(1 - \frac{y_1}{k_2} \right) \quad (\text{a})$$

式中, $(1 - y_1/k_2)$ 项描述由于营养限制而导致的生长停止。

青霉素产生速率为

$$\frac{dy_2}{dt} = k_3 y_1 - k_4 y_2 \quad (\text{b})$$

青霉素产生速率与细胞浓度 (y_1) 成正比, 随水解 (与青霉素浓度 y_2) 而减少。

程序说明 以加权残差平方和为优化目标函数, 用 `lsqnonlin()` 进行多变量非线性回归, 其中 `Func()` 定义目标函数。模型方程的初始条件取 $t = 0$ 时实验数据的平均值, 即 $y_1(0) = 0.29$ 和 $y_2(0) = 0.0$ 。权因子根据重复的实验数据计算, 见光盘中的程序 `weights.m`。由于模型是 ODE 方程, 可先用 `ode45()` 求解, 然后插值估算对应于自变量向量的因变量 (响应变量) 值。因输入数据包括重复的实验数据, 但 `ode45` 只处理上升或下降的自变量向量, 故用插值方法。

程序清单 *FermKineticsEst.m*

```
function FermKineticsEst
clear all; clc
load FermKineticsData      % 装载实验数据

% 排序
for i = 1:length(n);
    [x(1:n(i), i), index] = sort(x(1:n(i), i));
    y(1:n(i), i) = y(index, i);
end
w = Weights(x, y, n)

% 用 lsqnonlin() 进行非线性最小二乘参数估计 (非线性数据拟合)
beta0 = [0.1  4.0  0.02  0.02]; y0 = [0.29  0.0];
lb = [0  0.01  0  0]; ub = [inf  inf  inf  inf];
[beta, resnorm, residual, exitflag, output, lambda, jacobian] = ...
```

```

lsqnonlin(@Func, beta0, lb, ub, [], x, y, n, y0, w);
ci = nlparci(beta, residual, jacobian);
% 参数辨识结果
fprintf('\nEstimated Parameters by Lsqnonlin():\n')
fprintf('\tk1 = %.4f ± %.4f\n', beta(1), ci(1, 2) - beta(1))
fprintf('\tk2 = %.2f ± %.2f\n', beta(2), ci(2, 2) - beta(2))
fprintf('\tk3 = %.4f ± %.4f\n', beta(3), ci(3, 2) - beta(3))
fprintf('\tk4 = %.2f ± %.2f\n', beta(4), ci(4, 2) - beta(4))
fprintf(' The sum of the squares is: %.1e\n\n', sum(residual.^2))

% -----
function f = Func(beta, x, y, n, y0, w)
tspan = [0 max(max(x))];
[tt yy] = ode45(@ModelEqs, tspan, y0, [], beta);
for col = 1:length(n)
    yc(1:n(col), col) = spline(tt, yy(:, col), x(1:n(col), col));
end
f1 = y(:, 1) - yc(:, 1); f2 = y(:, 2) - yc(:, 2);
f = [sqrt(w(1))*f1; sqrt(w(2))*f2];

% -----
function dydt = ModelEqs(t, y, beta) % 模型方程
dydt = [beta(1)*y(1)*(1 - y(1)/beta(2)); beta(3)*y(1) - beta(4)*y(2)];

```

Weights.m (代码见光盘)

计算结果 $k_1 = 0.0432 \pm 0.0052$, $k_2 = 3.93 \pm 0.26$, $k_3 = 0.0174 \pm 0.0051$, $k_4 = 0.02 \pm 0.01$ 。
残差平方和: $2.3e+000$ 。

符号说明

k_1, k_2, k_3, k_4	发酵动力学参数	$y_{1 \text{ exp}}$	细胞浓度实验值, 单位/ml
t	发酵时间, h	y_2	青霉素的浓度, 单位/ml
y_1	细胞的浓度, 单位/ml	$y_{2 \text{ exp}}$	青霉素浓度实验值, 单位/ml

7.3.3 传热参数估计

【例 7-10】 化学热泵制冷系统的反应动力学参数和反应器传热参数的辨识^[18]

化学平衡方程

$\text{SrCl}_2/\text{NH}_3$ 可逆气-固化学反应式为: $\text{SrCl}_2 \cdot \text{NH}_3 + 7\text{NH}_3 \xrightleftharpoons[\text{解吸}]{\text{吸附}} \text{SrCl}_2 \cdot 8\text{NH}_3 + 7\Delta H_r$

$\text{SrCl}_2/\text{NH}_3$ 可逆气-固化学平衡遵循 Clapeyron 方程:

$$\ln p_{\text{eq}} = -\frac{4983.28}{T} + 27.5100 \quad (1)$$

动力学模型

$\text{SrCl}_2/\text{NH}_3$ 可逆气-固化学反应动力学方程为:

$$\frac{dx(t,r)}{dt} = k_{0a} \exp\left(-\frac{E_a}{RT(t,r)}\right) \cdot [1-x(t,r)]^{M_a} \cdot \frac{p_c - p_{ca}(T(t,r))}{p_{ca}(T(t,r))} \quad (\text{吸附}) \quad (2)$$

$$\frac{dx(t,r)}{dt} = k_{0d} \exp\left(-\frac{E_d}{RT(t,r)}\right) \cdot [1-x(t,r)]^{M_d} \cdot \frac{p_{cd}(T(t,r)) - p_c}{p_{cd}(T(t,r))} \quad (\text{解吸}) \quad (3)$$

式中, $x(t,r)$ 为局部反应进度; $T(t,r)$ 为反应器床层中的局部温度; $p_{ca}(T(t,r))$ 和 $p_{cd}(T(t,r))$ 都由式 (1) 定义, k_{0a} , E_a , M_a 及 k_{0d} , E_d , M_d 为待辨识别的动力学参数。

方程 (2) 和 (3) 的初始条件为:

$$t=0, x(0,r)=0 \quad (4)$$

总反应进度 $X_{gl}(t)$ 表示为:

$$X_{gl}(t) = \frac{1}{\pi(r_w^2 - r_g^2)} \int_{r_g}^{r_w} x(t,r) \cdot 2\pi r dr \quad (5)$$

将上式沿空间上积分得

$$X_{gl}(t_i) = \frac{1}{r_w^2 - r_g^2} \sum_{j=1}^{NX} [x(t_i, r_j)(r_j^2 - r_{j-1}^2)] \quad (6)$$

辨识动力学参数 k 、 E 和 M 的目标函数为:

$$\text{Min } J = \frac{1}{NX} \sum_{i=1}^{NX} \left\{ P_1 \left[\frac{X_{mes}(t_i) - X_{sim}(t_i)}{X_{mes}(t_i)} \right]^2 - P_2 \left[\frac{\dot{X}_{mes}(t_i) - \dot{X}_{sim}(t_i)}{\dot{X}_{mes}(t_i)} \right]^2 \right\} \quad (7)$$

式中, X 为总反应进度, \dot{X} 为总反应速率, 而 P_1 和 P_2 为权因子。

热量平衡方程

对反应介质,

$$c_p(x(t,r)) \frac{\partial T}{\partial t} = \frac{1}{r} \frac{\partial}{\partial r} \left[\lambda_e(x(t,r)) \cdot r \frac{\partial T}{\partial r} \right] + 7N_s \Delta H_r \cdot \frac{\partial x(t,r)}{\partial t} \quad (\text{吸附过程}) \quad (8)$$

$$c_p(x(t,r)) \frac{\partial T}{\partial t} = \frac{1}{r} \frac{\partial}{\partial r} \left[\lambda_e(x(t,r)) \cdot r \frac{\partial T}{\partial r} \right] - 7N_s \Delta H_r \cdot \frac{\partial x(t,r)}{\partial t} \quad (\text{解吸过程}) \quad (9)$$

$$\text{I.C.} \quad t=0, \begin{cases} T(0,r) = T_0(r) \\ x(0,r) = 0 \end{cases} \quad (10)$$

$$\text{B.C.} \quad r=r_g: \quad \left. \frac{\partial T}{\partial r} \right|_{r_g} = 0 \quad (11)$$

$$r=r_w: \quad -\lambda_e \left. \frac{\partial T}{\partial r} \right|_{r_w} = h_{sw} (T(r_w) - T_c) \quad (12)$$

式中, N_s 为单位反应介质的 SrCl_2 摩尔数 (这里为 2671 mol m^{-3})。

传热参数辨识的优化准则:

$$\begin{aligned} \text{Min } J = & \frac{1}{NX} \sum_{i=1}^{NX} \left\{ P_1 \left[\frac{X_{mes}(t_i) - X_{sim}(t_i)}{X_{mes}(t_i)} \right]^2 - P_2 \left[\frac{\dot{X}_{mes}(t_i) - \dot{X}_{sim}(t_i)}{\dot{X}_{mes}(t_i)} \right]^2 \right\} \\ & + \frac{1}{NT} \sum_{k=1}^{NT} \left\{ \frac{1}{Nr} \sum_{j=1}^{Nr} P_3 \left[\frac{T_{mes}(t_k, r_j) - T_{sim}(t_k, r_j)}{T_{mes}(t_k, r_j)} \right]^2 \right\} \end{aligned} \quad (13)$$

式中, N_X 为总反应进度测量值的实验点数, N_T 为温度测量值的实验点数, N_R 为用于径向位置温度的热电偶数目 (这里 $N_R = 6$), 而 P_1 、 P_2 和 P_3 是权因子。

动力学参数和传热参数的辨识

尽管在总体模型中, 动力学与传热相耦合, 但是两类不同参数是可以分别加以辨识的。

由于径向动态温度 $T(t, r)$ 已由实验测定, 因此可对动力学方程 (2) 和 (3) 进行数值积分得到局部反应进度 $x(t, r)$ 。这样, 可由 (6) 式求出在 t_i 时刻的总反应进度 $X_{gt}(t_i)$ 。因此, 通过比较总反应进度测量值与计算值, 使目标函数 (7) 式的函数值最小, 即可估计得到吸附/解吸的动力学参数。

在获得动力学参数以后, 可以求解热量平衡方程 (8)、(9) 以辨识传热参数。为简化起见, 有效导热系数和床层与反应器壁之间的传热系数可认为是常数。首先把方程 (8) 和 (9) 在整个时间和空间上离散, 并用有限差分进行数值求解。使用 MATLAB 的优化工具箱函数, 可搜索得到最优传热参数, 使得目标函数式 (10) 最小。

本例的程序较复杂, 这里从略。

7.3.4 相平衡参数估计

分离过程的数学模型一般包括气液相平衡 (VLE) 数据的非线性拟合。通常是, 利用最小二乘法拟合实验测得的 VLE 数据。对于双组分混合物, 通常用两个可调参数来关联 VLE 数据。在许多情况下, 只需要双组分参数即可以预测多组分气-液平衡。

下面只举简单的二元系统 VLE 数据拟合的例子。考虑逸度系数计算的二元参数估计, 参数估计方法完全一样, 只是增加逸度系数的计算过程, 这方面可参考 Raman (1985) [19]。

【例 7-11】估计二元混合物的 van Laar 模型中的参数^[20]

已知水-二氧杂环乙烷 (1,4-dioxane) 系统的 VLE 实验数据 (液相组成及总压), 如表 7-11。

表 7-11 水-二氧杂环乙烷系统的 VLE 实验数据

摩尔分数 x_1	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
$P^{sat}/\text{mmHg}^{①}$	28.10	34.40	36.70	36.90	36.80	36.70	36.50	35.40	32.90	27.70	17.50

① 1mmHg=133.322Pa, 下同。

水和二氧杂环乙烷的饱和蒸气压按照如下的 Antoine 方程计算:

$$\lg p^{\text{sat}} = a_1 - \frac{a_2}{T + a_3} \quad (\text{a})$$

式中, p^{sat} 为饱和蒸气压, mmHg; T 为温度, $^{\circ}\text{C}$; a_1, a_2, a_3 为 Antoine 常数 (表 7-12)。

表 7-12 Antoine 常数

组 分	a_1	a_2	a_3	使用范围
水	8.07131	1730.630	233.426	1~100 $^{\circ}\text{C}$
二氧杂环乙烷	7.43155	1554.679	240.337	20~105 $^{\circ}\text{C}$

试利用已知的 VLE 实验数据估计 van Laar 模型中的未知参数, 并计算对应的气相摩尔分数。

数学模型

对于低压系统, 有 $p_i = y_i p = x_i \gamma_i p_i^{\text{sat}} \quad (i = 1, 2)$ (b)

总压为 $p = \sum_i p_i$ (c)

式中, p 为总压; p_i^{sat} 为组分 i 的饱和蒸气压; p_i 为组分 i 的分压; x_i 为组分 i 的液相摩尔

分数: y_i 为组分 i 的气相摩尔分数; γ_i = 活度系数。

$$\text{二元混合物的 van Laar 模型为} \quad \ln \gamma_1 = A_{12} \left(\frac{A_{21} x_2}{A_{12} x_1 + A_{21} x_2} \right)^2 \quad (\text{d})$$

$$\ln \gamma_2 = A_{21} \left(\frac{A_{12} x_1}{A_{12} x_1 + A_{21} x_2} \right)^2 \quad (\text{e})$$

式中, A_{12} 和 A_{21} 是需要估计的二元参数。

因此, 对于给定温度的二元系统 ($x_2 = 1 - x_1$), 将 (a)、(d) 和 (e) 代入 (b) 得到 p_i , 再代入 (c) 即可得到:

$$p = p(x_1, A_{12}, A_{21}) \quad (\text{f})$$

即由 (f) 式得到对应于 x_1 的总压计算值 p_j^{cal} 。

$$\text{然后, 根据优化目标} \quad \min f = \sum_j^n (p_j^{\text{cal}} - p_j^{\text{exp}})^2 \quad (\text{g})$$

利用非线性最小二乘法即可估计得到参数 A_{12} 和 A_{21} 。

程序说明 ObjFunc() 定义目标函数式 (g), Psat() 根据 (a) 式计算饱和蒸气压, PressureCal() 则根据式 (d)、(e)、(b)、(c) 可计算组分 1 和组分 2 的分压 p_1 、 p_2 以及总压 p^{cal} 。

程序采用 lsqnonlin() 求解, 其中初值根据经验取 $\text{beta0} = [A_{12} \ A_{21}] = [1 \ 1]$ 。具体计算时, 读者可变换不同的初值, 如 $\text{beta0} = [5 \ 5]$, 重复计算过程, 将发现, 程序收敛于同一结果。

程序清单 见光盘中的 *VLEfitting.m*。

计算结果 ① 得到的参数估值为: $A_{12} = 1.958 \pm 0.069$, $A_{21} = 1.689 \pm 0.059$; ② 对应于 x_1 的气相摩尔分数 y_1 为: $y_1 = [0.0000 \ 0.2345 \ 0.3111 \ 0.3408 \ 0.3534 \ 0.3618 \ 0.3740 \ 0.3987 \ 0.4512 \ 0.5757 \ 1.0000]$; ③ 计算得到的 y_1 - x_1 关系, 如图 7-6。

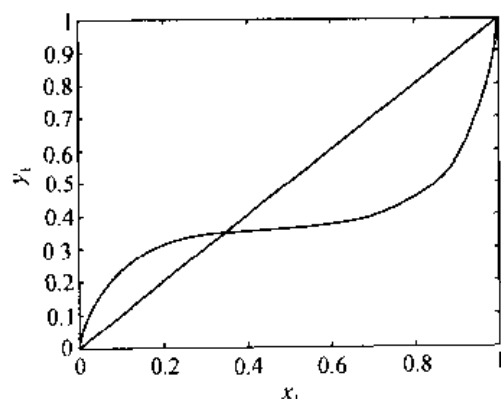


图 7-6 汽液平衡曲线

由图 7-6 可见, 该系统出现恒沸点 $y_1 = x_1 = 0.35$ 。

7.4 模型辨识

模型辨识的任务在于竞争模型的剔除和判别, 即用统计方法从多个可能的模型中筛选出能与实验数据拟合得最好的最佳模型。具体地说, 竞争模型间的判别, 可以参数必须为正值和拟合优度为基础,

并辅以统计检验。因此, 模型辨识实际上是参数估计和统计检验的结合。

参数是否为正及拟合优度如何, 属于上一节的参数估计问题。下面介绍有关统计检验。

模型辨识的几个重要参数

为了判别和剔除模型, 必须考虑以下几个重要参数:

(1) 决定性指标 ρ^2 : 用于度量对回归均值总偏离的大小。

$$\rho^2 = 1 - \frac{\sum_{i=1}^{N_e} (y_{ei} - y_{ci})^2}{\sum_{i=1}^{N_e} y_{ei}^2} \quad (7-15)$$

式中, N_e 为实验点数目; y_{ei} 、 y_{ci} 为第 i 次观测的因变量的实验值和计算值。显然, ρ^2 值越大越好。

(2) F 比 回归均方和与模型计算误差均方和之间的比值。

$$F = \frac{\left[\sum_{i=1}^{N_e} y_{ei}^2 - \sum_{i=1}^{N_e} (y_{ei} - y_{ci})^2 \right] / N_p}{\sum_{i=1}^{N_e} (y_{ei} - y_{ci})^2 / (N_e - N_p)} \quad (7-16)$$

式中, N_p 、 $(N_e - N_p)$ 分别是上述两项方和的自由度 (N_p 为模型参数的维数)。

根据 F 比值, 可判断是否存在对回归均值的偏差。显然, F 比值越大越好, 当 $F > 10F_t$ (F_t 为 F 检验表的值) 时, 表明模型可以较好地预示实验结果。 F_t 即 $F_{\alpha}(N_p, N_e - N_p)$ 一般取显著水平为 $\alpha = 5\%$ 、相应自由度 N_p 和 $(N_e - N_p)$ 下的 F 表值, 可查有关数理统计的书籍。

(3) F_E 比 模型计算误差均方与实验误差方差之比。

$$F_E = \frac{\sum_{i=1}^{N_e} (y_{ei} - y_{ci})^2 / (N_e - N_p)}{\sigma_E^2} = \frac{\sigma^2}{\sigma_E^2} \quad (7-17)$$

可根据此值判断模型计算误差方差是否明显大于实验误差方差, 从而给出模型的绝对优劣。

(4) 第 i 个模型的后验概率 π_i , 据此可对模型预示附加实验的能力作出估计。

在模型辨识过程中, 通过对统计参数 ρ^2 、 F 、 F_E 和 π_i 值的综合分析, 可判别模型的优劣。

决定性指标 ρ^2 主要给出关于各模型性质相对比较的信息, 即不论哪个模型, 如果它的 ρ^2 值比其他模型的 ρ^2 值低得多, 就可以认为它是不太令人满意的。

可将各模型的 F 值与 F 检验表中相应的值加以比较。Drapper 和 Smith (1966)^[21] 认为, 在做 F 检验时, 即使对于线性回归, 也必须采用一个安全系数。只有这样, 模型才能令人满意地预示实验结果。也就是说, 各模型的 F 值与 F 表值乘上相应的安全系数后所得之值加以比较。他们建议的线性回归的安全系数取为 4, 非线性回归的安全系数取为 10。

同样, 可将各模型的 F_E 值与 F 检验表中相应的值加以比较。在比较时, 非线性回归的安全系数也取为 10。

最后, 后验概率对模型可靠性给出综合的评价。后验概率的估算方法可参考文献[4]。

绘制残差分析图

为了协助判别和剔除模型, 绘制残差分析图是很有帮助的。残差分析图的绘制方法同上一节, 即将各次实验的残差 ($y_{ei} - y_{ci}$) 对 y_{ei} 标绘, 检查是否均匀地分布在零线的上下侧。

还可以根据下面介绍的方法定量地判断残差的随机性, 即比较模型之间的残差性态。

残差分布的随机性检验 (the randomness test)^[17]

假设正、负残差值的个数分别为 n_1 和 n_2 , 残差正负符号改变的次数为 r , r 的分布近似为正态分布。此变量的均值和标准差为

$$\bar{r} = \frac{2n_1n_2}{n_1 + n_2} + 1 \quad (7-18)$$

$$\sigma = \sqrt{\frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}} \quad (7-19)$$

模型辨识简便方法

为简便起见, 如果 $\rho^2 > 0.9$, $F > 10F_{0.05}(N_p, N_e - N_p)$, 且残差分析均匀地分布在对角线两侧, 或均匀地分布在零线的上下侧, 则可以认为所选模型和参数估值是可靠的。

【例 7-12】请判断[例 7-4]的模型是否适定

程序说明 要判断[例 7-4]的模型是否适定, 需计算 ρ^2 和 F 。现根据式 (7-15) 和 (7-16) 编写计算 ρ^2 和 F 的通用函数 rho2_F(), 并作为外部函数单独保存为 rho2_F.m。在例 7-4 的程序的基础上, 只需增加调用 rho2_F() 以计算 ρ^2 和 F 的代码即可, 见光盘中的程序 KineticsEst_Ident1.m。

计算结果 运行程序 KineticsEst_Ident1.m 得到, 实验点数和自由度分别为 $N_e = 12$ 和 $N_p = 3$, 决定性指标 $\rho^2 = 0.998$, $F = 1645.361$ 。其他结果同[例 7-4]。查 F 表知, $F_{0.05}(N_p, N_e - N_p) = F_{0.05}(3, 9) = 3.86$ 。可见, $\rho^2 > 0.9$, $F > 10 F_{0.05}(3, 9)$, 且由图 7-4 残差图看出, 残差的性态很好。因此, 该模型是适定的。

【例 7-13】在[例 7-2]中, 给定动力学实验数据 (表 7-3), 估计动力学模型

$$-r_A = \frac{k p_E p_H}{1 + K_E p_E} \quad (a)$$

中的参数 k 和 K_E 。现在[例 7-2]的基础上, 增加另外三个动力学模型 (b)、(c) 和 (d):

$$-r_A = \frac{k p_E p_H}{1 + K_A p_{EA} + K_E p_E} \quad (b)$$

$$-r_A = \frac{k p_E p_H}{(1 + K_E p_E)^2} \quad (c)$$

$$-r_A = k p_E^a p_H^b \quad (d)$$

试根据表 7-3 中数据确定在以上四个模型中哪个模型最好。

程序说明 用 lsqnonlin() 进行参数估计, 其中 ObjFunc() 定义目标函数。通过标志变量 IDmodel 的值 'a'、'b'、'c' 或 'd' 选择相应模型 (a)、(b)、(c) 或 (d)。模型 (a)、(b)、(c)、(d) 的残差见图 7-7。

程序清单 见光盘中的 KineticsEst_Ident2.m。

计算结果

模 型	参 数	收敛值	0.95%置信区间	下 限	上 限	残差平方和
(a)	k	3.19	0.288	2.899	3.475	0.042
	K_E	2.101	0.263	1.837	2.365	
(b)	k	3.35	0.391	2.957	3.739	0.030
	K_F	2.21	0.319	1.892	2.530	
	K_A	0.043	0.071	-0.029	0.114	
(c)	k	2.009	0.266	1.743	2.275	0.436
	K_F	0.362	0.062	0.299	0.424	
(d)	k	0.894	0.257	0.637	1.151	0.297
	a	0.258	0.071	0.188	0.329	
	b	1.061	0.209	0.852	1.271	

根据计算结果比较: 模型 (b) 的残差平方和最小, 但 $K_A = 0.043 \pm 0.071$, 其 95% 的置信区间比参数本身还大, 因此造成参数 K_A 的值为负 (下限为负), 这从物理意义来说是不可能的。故该模型不可取。

注意: 不能简单地根据最小的残差平方和来筛选模型!

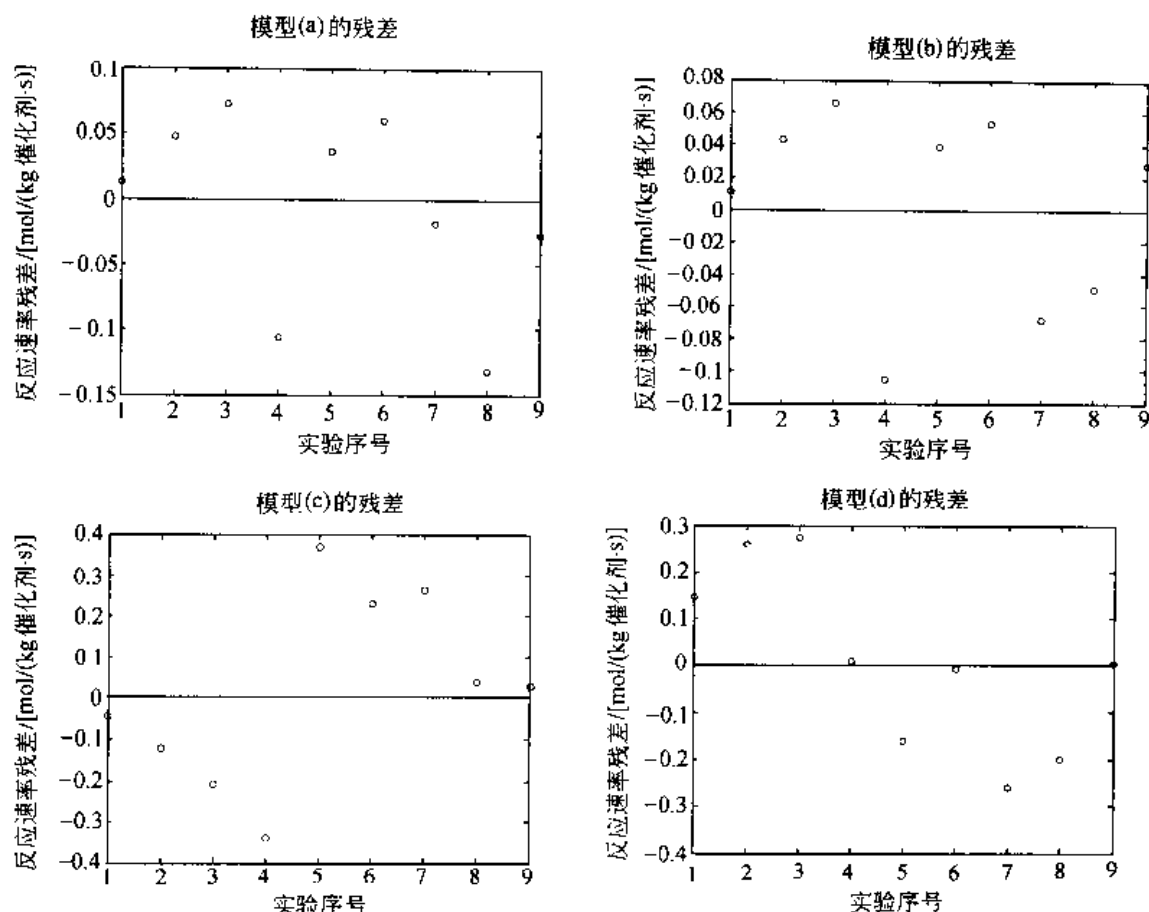


图 7-7 残差

习 题

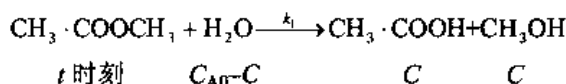
7-1 在实验室间歇反应器中进行醋酸甲酯水解反应，其中正反应为拟一级反应，逆反应为二级反应。开始时只有醋酸甲酯溶液而没有任何产物，醋酸甲酯的初始浓度 C_{A0} 为 0.05 kmol/m^3 ，不同时刻的水解转化率如下表所示。反应式 $\text{CH}_3 \cdot \text{COOCH}_3 + \text{H}_2\text{O} \xrightarrow{k_1} \text{CH}_3 \cdot \text{COOH} + \text{CH}_3\text{OH}$ ，实验结果如下表。

时间/s	0	1350	3060	5340	7740	∞
转化率 x	0	0.21	0.43	0.60	0.73	0.90

试估计正、逆反应速率常数 k_f 和 k_r [16]。

已知数学模型：

设 t 时刻每单位体积反应混合物被转化掉的醋酸甲酯物质的量（摩尔数）为 C ，各组分的浓度如下：



故醋酸甲酯的反应速率方程为 $r = \frac{dC}{dt} = k_f(C_{A0} - C) - k_r C^2$ (1)

平衡时 $C = C_e$, $dC/dt = 0$, 则 $K = \frac{k_1}{k_r} = \frac{C_e^2}{C_0 - C_e}$ (2)

以 k_f/K 代替 (1) 式中的 k_r , 得 $\frac{dC}{dt} = k_f(C_{A0} - C) - \frac{k_1}{K} C^2$ (3)

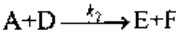
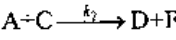
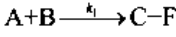
7-2 用硅藻土作载体的镍催化剂，进行苯加氢合成环己烷。用微分反应管测定 160°C 的初始反应速率 r_0 ，

结果如表 7-13 所示。初始反应速率表达式为： $r_0 = \frac{k K_H^3 K_B p_H^3 p_B}{(1 + K_H p_H + K_B p_B)^4}$ ，试辨识动力学参数 k ， K_H ， K_B ^[22]。

表 7-13 苯加氢合成环己烷的动力学实验数据（单位： p_H 、 p_B 为 atm， r_0 为 mol/(g·h)）

p_H	0.7494	0.6721	0.5776	0.5075	0.9256	0.9266	0.8766	0.7564	0.5617	0.5241
p_B	0.2670	0.3424	0.4342	0.5043	0.1020	0.0997	0.1471	0.2607	0.4501	0.4877
r_0	0.2182	0.2208	0.2235	0.1892	0.1176	0.1151	0.1472	0.2178	0.2122	0.2024

7-3 对于反应系统



得到微分方程组

$$\frac{dC_A}{dt} = -k_1 C_A C_B - k_2 C_A C_C - k_3 C_A C_D$$

$$\frac{dC_B}{dt} = -k_1 C_A C_B$$

$$\frac{dC_C}{dt} = k_1 C_A C_B - k_2 C_A C_C$$

$$\frac{dC_D}{dt} = k_2 C_A C_C - k_3 C_A C_D$$

$$\frac{dC_E}{dt} = k_3 C_A C_D$$

已知初始条件为 $C_C(0) = C_D(0) = 0.0$ ， $C_A(0) = 0.02090$ mol/L， $C_B(0) = C_A(0)/3$ ，实验数据如表 7-14 所示。试估计参数 k_1 、 k_2 和 k_3 ^[23]。

表 7-14 实验数据

时间/min	$C_A \times 10^3 / (\text{mol/L})$	时间/min	$C_A \times 10^3 / (\text{mol/L})$	时间/min	$C_A \times 10^3 / (\text{mol/L})$
4.50	15.40	46.33	9.780	147.92	6.646
8.67	14.22	57.00	9.157	198.00	5.883
12.67	13.35	69.00	8.594	241.75	5.322
17.75	12.32	76.75	8.395	270.25	4.960
22.67	11.81	90.00	7.891	326.25	4.518
27.08	11.39	102.00	7.510	418.00	4.075
32.00	10.92	108.00	7.370	501.00	3.715
36.00	10.54				

问题来源：Cutlip (1999, P.526)^[23]引自 D. M. Himmelblau, Process Analysis by Statistical Methods, Wiley, New York, 1970，原始文献为 Svirebely (1961)^[23]。

7-4 模型辨识^[13]

Cutlip 和 Peters 研究了叔丁醇 (A) 脱水为异丁烯 (B) 和水 (C) 的非均相催化反应，该反应是在常压及各种温度下进行研究。在 533.1 K 温度下改变不同的反应物和产物的分压，测得的反应速率如表 7-15。

表 7-15 533.1 K 温度下的动力学数据

No.	分压/atm			r_A /[mol/(h·g)]
	p_A	p_B	p_C	
1	0.7913	0.0172	0.0177	0.005047
2	0.6349	0.0156	0.0159	0.004409
3	0.4788	0.0146	0.0149	0.003857
4	0.3339	0.0163	0.0157	0.003048
5	0.6362	0.1736	0.0146	0.004464

续表

No	分压/atm			r_A /[mol/(h·g)]
	p_A	p_B	p_C	
6	0.4864	0.3252	0.0128	0.003671
7	0.3302	0.4819	0.0135	0.002716
8	0.651	0.0104	0.1629	0.004271
9	0.474	0.0122	0.3374	0.003244
10	0.3167	0.01	0.4982	0.002348
11	0.3506	0.314	0.0121	0.002841
12	0.3973	0.0121	0.2705	0.002903
13	0.3661	0.0083	0.186	0.002995
14	0.3219	0.1819	0.0117	0.002801
15	0.4737	0.1821	0.0135	0.003622
16	0.4857	0.0089	0.1687	0.003523

考虑许多不同的反应机理（具有不同的速率控制步骤），可导出 24 种关于分压的反应速率模型。经过初始筛选后，余以下四种模型需要进一步辨识：

$$\text{模型 a} \quad r_A = \frac{kK_A p_A}{1 + K_A p_A + K_C p_C} \quad (\text{a})$$

$$\text{模型 b} \quad r_A = \frac{kK_A p_A}{1 + K_A p_A + K_B p_B + K_C p_C} \quad (\text{b})$$

$$\text{模型 c} \quad r_A = \frac{kK_A p_A}{(1 + K_A p_A + K_C p_C)^2} \quad (\text{c})$$

$$\text{模型 d} \quad r_A = \frac{kK_A p_A}{(1 + K_A p_A + K_B p_B + K_C p_C)^2} \quad (\text{d})$$

式中， r_A 为反应速率，mol/(h·g)； k 为反应速率常数，mol/(h·g)； p_A 、 p_B 和 p_C 分别是叔丁醇、异丁烯和水的分压，atm； K_A 、 K_B 和 K_C 分别是叔丁醇、异丁烯和水的吸附常数，atm⁻¹。

Cutlip 和 Peters 还考虑如下的一个经验的指数型速率模型：

$$\text{模型 e} \quad r = k p_A^a p_B^b p_C^c \quad (\text{e})$$

式中， a 、 b 和 c 为未知指数。

参 考 文 献

- 1 邓正龙主编. 化工中的优化方法. 北京: 化学工业出版社, 1992, 58
- 2 John H. Seinfeld and Leon Lapidus. Mathematical Methods in Chemical Engineering Volume 3 Process Modeling, Estimation, and Identification. New Jersey: Prentice-Hall Inc., 1974.
- 3 Englezos & Kalogerakis. Applied Parameter Estimation for Chemical Engineers. Marcel Dekker, 2000 (16.1.3)
- 4 化学工业部化工科研计算机应用中心站译. 序贯实验设计方法译文集. 北京: 化学工业出版社, 1983
- 5 Douglas M. Bates and Donald G. Watts, Nonlinear regression analysis and its applications. John Wiley & Sons, 1988. 中译本: 韦博成, 万方焕, 朱宏图, 张绕庭, 非线性回归分析及其应用. 第一版. 北京: 中国统计出版社, 1997.11
- 6 Smith, J. M., Chemical Engineering Kinetics. 3rd edition. McGraw-Hill, 1981 有中译本, 王建华, 许学书, 黄世英, 刘栋昌和江礼科译. 化工动力学 (第三版). 化学工业出版社和成都科技出版社, 1988
- 7 Octave Levenspiel, Chemical Reaction Engineering. Third Edition. John Wiley & Sons, Inc. 1999, 60~61
- 8 Froment, G. F., and Bischoff, K. B. Chemical Reactor Analysis and Design. 2nd edition. Wiley, 1990.
- 9 Fogler H. S. Elements of Chemical Reaction Engineering. 3rd Edition. Prentice Hall PTR, Upper Saddle River, New Jersey, 1999
- 10 朱炳辰主编. 化学反应工程. 北京: 化学工业出版社, 2001
- 11 朱开宏, 袁渭康编著. 化学反应工程分析. 北京: 高等教育出版社, 2002
- 12 Kittrell J. R. Mathematical Modeling of Chemical Reactions. Advan Chem Eng, 1970
- 13 M. B. Cutlip and M. Shacham, Problem Solving in Chemical Engineering, 1999, 8.10

- 14 Ferich Keil, et al. ed., Scientific computing in chemical engineering II, 1999, p.351
- 15 Belohlav Z., Zamostny P., Kluson P. and Volf J. Application of Random-Search Algorithm for Regression Analysis of Catalytic Hydrogenations. The Canadian Journal of Chemical Engineering, V.75, 1997, 738~739
- 16 Ricardson J. F. and Peacock D. G., Chemical Engineering, Vol.3, Chemical & Biochemical Reactor & Process Control. 3rd edition. 世界图书出版社, 1994
- 17 Alkis Constantinides & Navid Mostoufi. Numerical Methods for Chemical Engineers with MATLAB Applications. Prentice Hall, 1999
- 18 Hua-Jiang Huang, Guo-Bin Wu, Jiao Yang, Ying-Chun Dai*, Wei-Kang Yuan, and Hui-Bo Lu. Modeling of Gas-solid Chemisorption in Chemical Heat Pumps. SEPERATION AND PURIFICATION TECHNOLOGY. 2003. (in press)
- 19 Raman R., Chemical Process Computation. London: Elsevier Applied Science Publishers, 1985. 中译本, 许锡恩, 张福芝, 王保国, 吴诗华译. 化工过程计算. 北京: 化学工业出版社, 1992
- 20 Edgar, T. F., Himmelblau D. M. and Lasdon L. S., Optimization of Chemical Processes. 2nd edition. McGraw-Hill, 2001, 451
- 21 N. R. Drapper and H. Smith, Applied Regression Analysis, J. Wiley, New York, 1966
- 22 王树森编. 化学工程计算方法. 北京: 化学工业出版社, 1989, 326
- 23 Svirbely, W. J. and Blaner, J. A., J. Amer. Chem. Soc., vol. 83, 1961, 4118
- 24 Statistics Toolbox User's Guide. Version 4.0. The MathWorks, Inc., 2002 (stats_tb.pdf)
- 25 Optimization Toolbox User's Guide. Version 2.2. The MathWorks, Inc., 2002 (optim_tb.pdf)

第 8 章 化工试验设计及数据处理

8.1 概述

科学实验与研究是花费大量人力、物力的工作，比如化学化工、生化医药、材料等领域的科学实验与研究开发，常常需要进行许多次的多因素实验，以寻找最佳操作条件或最佳配方。在模型化方面，也需要进行实验以估计模型中的未知参数。因此，如何用更科学的方法有效地安排多因素实验，对提高科研效率和获取满意的研究结果至关重要。

实验设计 (experimental design) 是以概率论、数理统计以及最优化方法等为理论基础，进行科学合理的实验安排。

8.1.1 基本概念

试验指标 试验需要考虑的结果称为试验指标 (简称指标)，如转化率、收率、产品性能、产品质量、成本、利润等均可作为衡量试验效果的指标。

试验因素 对试验指标有影响的变量，用大写字母 A、B、C…表示。例如，以转化率为试验指标，温度、压力和流量等操作条件对转化率有影响，则温度、压力和流量是三个因素。

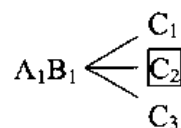
因素的水平 每个因素可能处在的状态称为因素的水平 (简称本平)。

8.1.2 试验设计方法

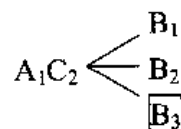
试验设计方法主要有四种：全面试验法、简单比较法、正交试验法、序贯试验法等。其中，全面试验、简单比较试验和正交试验都属于析因实验设计，也就是先验型的设计。而序贯试验法的优点是可以利用前面的实验中得到的信息来设计随后的实验。

全面试验法 (完全析因设计)：即把全部因素和水平按排列组合的方式进行全面试验。例如，三因素三水平的全面试验法需做 $3^3=27$ 次试验。这种方法虽然有全面的优点，但试验次数太多，很费时。

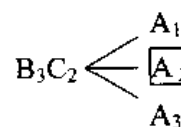
简单比较法：即每次试验只改变一个因素，而固定其他因素，如 A、B、C 三因素三水平试验，首先固定 A_1B_1 ，使 C 变化，则



如 $A_1B_1C_2$ 的结果最好，则固定 A_1C_2 ，使 B 变化，则



如 $A_1B_3C_2$ 的结果最好，则固定 B_3C_2 ，使 A 变化，则



最后，如果 $A_2B_3C_2$ 的结果最好，则认为最好的工艺条件是 $A_2B_3C_2$ 。

这种方法一般也很有效果。而且试验次数比全面试验法少得多，其主要缺点是：(1) 选点代表性很差，同样的试验次数，提供的信息不够丰富；(2) 无法考察因素间的交互作用。

正交试验法：所谓正交试验法就是用正交表安排试验方案，它可以解决多因素、多水平及多指标这一类的试验问题，详见 8.3。

序贯试验法：实际上是最优化方法与统计理论在实验规划中的应用，详见 8.5。

8.2 常用数理统计

8.2.1 总体和样本的统计量

(1) 数学期望和均值

总体均值 μ (或数学期望 $E(X)$):

设总体 X 是连续型随机变量，密度函数为 $p(x)$ ，则总体均值 μ (或数学期望 $E(X)$) 为

$$\mu = E(X) = \int_{-\infty}^{+\infty} xp(x)dx \quad (8-1)$$

设总体 X 是离散型随机变量，概率分布是： $P\{X = x_k\} = p_k \quad (k = 1, 2, \dots)$ (8-2)

则总体 X 的均值 μ (或数学期望 $E(X)$) 为： $\mu = E(X) = \sum_k x_k p_k$ (8-3)

样本均值 \bar{x} :
$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (8-4)$$

相应的 MATLAB 函数为 `mean(x)`：计算向量 x 的算术平均值。若 x 为矩阵，则计算返回一向量，该向量包括 x 每列的算术平均值。

(2) 方差和标准差

总体方差 σ^2 和总体标准差 σ :

设总体 X 是连续型随机变量，密度函数为 $p(x)$ ，则总体方差 σ^2 为

$$\begin{aligned} \sigma^2 &= Var(X) \\ &= E((X - E(X))^2) = E((X - \mu)^2) \\ &= \int_{-\infty}^{+\infty} (x - \mu)^2 p(x)dx \end{aligned} \quad (8-5)$$

设总体 X 是离散型随机变量，概率分布是： $P\{X = x_k\} = p_k \quad (k = 1, 2, \dots)$ (8-6)

则总体方差 σ^2 为
$$\sigma^2 = \sum_k (x_k - \mu)^2 p_k \quad (8-7)$$

总体标准差 σ :
$$\sigma = +\sqrt{\sigma^2} \quad (8-8)$$

样本方差 s^2 和样本标准差 s :
$$s^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \quad (8-9)$$

因总体均值通常未知，常把样本均值作为总体均值的估计值，故样本方差由下式计算：

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (8-10)$$

样本标准差 s :
$$s = +\sqrt{s^2} \quad (8-11)$$

MATLAB 的样本标准差 s 计算函数为 `std(x)`。

(3) 协方差

两个随机变量 X 和 Y 的协方差 $Cov(X, Y)$ 定义为

$$Cov(X, Y) = E((X - E(X))(Y - E(Y))) \quad (8-12)$$

MATLAB 的协方差计算函数为 $Cov(x, y)$ 。

(4) 相关系数

两个随机变量 X 和 Y 之间的相关系数为: $\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{Var(X) Var(Y)}}$ (8-13)

式中, Var 为方差。

如果 $\rho_{XY} = 0$, 则称 X 和 Y 无关, 即 $Cov(X, Y) = 0$ 。如果 X 和 Y 是自变量, 可以证明 $Cov(X, Y) = 0$, 即 $\rho_{XY} = 0$, 因此, 自变量 (独立变量) 之间是无关的。

MATLAB 的相关系数计算函数为 $corrcoef(x, y)$ 。

现把上述总体和样本的统计量归结于表 8-1 中。

表 8-1 总体和样本的统计量小结

统计量	总 体		样 本
	连续变量	离散变量	
均 值	$\mu = E(X) = \int_{-\infty}^{\infty} xp(x)dx$	$\mu = E(X) = \sum_k x_k p_k$	$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$
方 差	$\sigma^2 = Var(X)$ $= E((X - E(X))^2)$ $= \int_{-\infty}^{\infty} (x - \mu)^2 p(x)dx$	$\sigma^2 = Var(X)$ $= E((X - E(X))^2)$ $= \sum_k (x_k - \mu)^2 p_k$	$s^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$ $s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$
标 准 差	$\sigma = +\sqrt{\sigma^2}$	$\sigma = +\sqrt{\sigma^2}$	$s = +\sqrt{s^2}$
协 方 差	$Cov(X, Y) = E((X - E(X))(Y - E(Y)))$		
相关系数	$\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{Var(X) Var(Y)}}$		

8.2.2 方差分析中的几个数学概念

(1) 偏差平方和 即一组数中各个数与它们的算术平均值之差的平方和, 以 S 表示。

设有 n 个数 y_1, y_2, \dots, y_n , 用 \bar{y} 表示它们的算术平均值, 即:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad (8-14)$$

则偏差平方和为

$$S = \sum_{i=1}^n (y_i - \bar{y})^2 \quad (8-15)$$

偏差平方和反映了一组数据的分散和集中程度。

(2) 平均偏差平方 为了合理地比较两组个数不同的数据的分散和集中程度, 应采用平均偏差平方和 (简称均方和)

$$\frac{S}{n-1} = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2 \quad (8-16)$$

式中, $(n-1)$ 就是 S 的自由度。

(3) F 比 因子水平的改变引起的平均偏差平方和与误差的平均偏差平方和的比值称为

$$F = \frac{S_{\text{因}}/f_{\text{因}}}{S_{\text{误}}/f_{\text{误}}} \quad (8-17)$$

F 比, 即

式中, $f_{\text{因}}$ 为因素的自由度: $f_{\text{因}} = \text{因素的水平数} - 1$ 。

(4) F 分布表及其查法 为了判断 F 比大小的意义, 即 F 比多大时可认为试验结果的差异主要是由因子水平的改变所引起的, 小到多少, 可以认为试验结果的差异主要是由试验误差所引起的, 这就需要有一个标准, 据以衡量 F 比, 这个标准就是根据统计数学原理编制的

F 分布表。 F 分布表列出了各种自由度情况下 F 比的临界值。在 F 分布表（见附录）上，行 $n_1: 1, 2, 3, 4, \dots$ 代表 F 比中分子的自由度，列 $n_2: 1, 2, 3, 4, \dots$ 代表 F 比中分母的自由度，表中的数值即各种自由度情况下 F 比的临界值。

8.3 正交实验设计与数据处理

8.3.1 正交实验设计方法

设 A 是一个 $n \times k$ 矩阵，其中第 j 列元素由 $1, 2, \dots, m$ ($j = 1, 2, \dots, k$) 构成，若 A 矩阵的任意两列搭配均衡，则称 A 是一张正交表。

通常把正交表记为 $L_n(m_1 \times m_2 \times \dots \times m_k)$ ，其中 L 是正交表的代号，是 Latin Square（拉丁方）的第 1 个字母； n 表示有 n 行（做 n 次试验）； $m_1 \times m_2 \times \dots \times m_k$ 表示共有数列，每列的水平数分别为 m_1, m_2, \dots, m_k 。

最简单的正交表是 $L_4(2^3)$ ，见表 8-2。

表 8-2 $L_4(2^3)$ 因素-水平

列号	1	2	3	列号	1	2	3
水平				水平			
1	1	1	1	3	2	1	2
2	1	2	2	4	2	2	1

先说一下记号 $L_4(2^3)$ 的含意。“ L ”代表正交表， L 下角的数字“4”表示有 4 行，即要做四次试验；括号内的指数“3”表示有 3 列，即最多允许安排的因子个数是 3 个；括号内的数“2”表示表的主要部分只有 2 种数字，即因子有两种水平 1 与 2，称之为 1 水平与 2 水平。

正交表 $L_4(2^3)$ 的两个特性：

(1) 每一列中，不同的数字出现的次数相等。这里不同的数字只有 1 和 2，各出现 2 次；

(2) 任意两列中，将同一行的两个数字看成有序数对（即左边的数放在前，右边的数放在后，按这一次序排出的数对）时，每种数对出现的次数相等。这里序数对共有四种：(1, 1), (1, 2), (2, 1), (2, 2)，它们各出现一次。

凡满足上述两个性质的表就称为正交表。

常见的正交表有： $L_4(2^3)$ ， $L_8(2^7)$ ， $L_{16}(2^{15})$ ， $L_4(2^3)$ ， \dots ； $L_9(3^4)$ ， $L_{27}(2^{13})$ ， \dots ； $L_{16}(4^5)$ ， \dots ； $L_{25}(5^6)$ ， \dots ，等。具体表格见光盘中的文档“正交表.doc”。

正交试验后得到的数据，要进行分析。数据分析方法主要有极差分析（直观分析法）和方差分析。下面将结合例子介绍使用正交表进行正交试验设计以及数据分析的方法。

8.3.2 正交实验结果的极差分析

8.3.2.1 无交互作用的正交实验

【例 8-1】某化工产品的收率 (Y) 与以下三个因素有关：反应温度 T ($^{\circ}\text{C}$)、反应压力 p (kPa)、催化剂种类 C 。因素-水平见表 8-3。

表 8-3 因素-水平

因素	温度 $T/^{\circ}\text{C}$	压力 p/kPa	催化剂 C
水平			
1	80	150	C_1
2	85	200	C_2
3	90	250	C_3

对这三个因素，如何安排正交试验？

按下面步骤安排正交试验及试验结果分析。

(1) 制定因素-水平表 (这里题目已给出);

(2) 选用合适的正交表 $L_n(r^m)$ 选择的原则是: r 要等于因素的水平数, m 要大于或等于因素的个数, n 为试验次数, 应尽可能小。本例是三因素三水平试验, 应选用 $L_9(3^4)$ 正交表;

(3) 设计表头 将各因素安排在正交表的各列上方, 每个因素占 1 列, 这称为表头。当不考虑因素之间的交互作用时, 表头上的因素可以任意安放。表头上不放因素的列称为空白列。本例把 T 、 P 和 C 三个因素放在 $L_9(3^4)$ 表的任意三列的表头上 (不考虑因素之间的交互作用);

(4) 确定试验方案, 并进行试验, 取得试验观测值 填入因素水平的具体内容, 即把 T 、 p 和 C 对应三列中的 “1”、“2”、“3” 转换为相应的具体水平。正交表的每一行代表一种水平组合, 对每一种水平组合做一次试验。正交表有 n 行, 即需做 n 次试验, 共得到 n 个试验观测值: Y_1, Y_2, \dots, Y_n 。本例 $n=9$, 正交试验方案与结果如表 8-4 所示, 其中, 结果 K 、 k 和 R 由程序 OrthExpAnalysis1.m 计算得到;

表 8-4 正交试验方案与结果

因素(列号)	T	p	C	收率 Y
试验号				
1	1 (80)	1 (150)	1 (C_1)	31%
2	1 (80)	2 (200)	2 (C_2)	54%
3	1 (80)	3 (250)	3 (C_3)	38%
4	2 (85)	1 (150)	2 (C_2)	53%
5	2 (85)	2 (200)	3 (C_3)	49%
6	2 (85)	3 (250)	1 (C_1)	42%
7	3 (90)	1 (150)	3 (C_3)	57%
8	3 (90)	2 (200)	1 (C_1)	62%
9	3 (90)	3 (250)	2 (C_2)	64%
K	123	141	135	
	144	165	171	
	183	144	144	
k	41	47	45	
	48	55	57	
	61	48	48	
R	20	8	12	

(5) 程序清单 OrthExpAnalysis1.m

```
function OrthExpAnalysis1 % 正交试验的极差分析
NL=3; NF=3; r=3; % NL: 水平数, NF: 因素数, 每个水平有 r 次试验
DATA = ... % DATA 矩阵的前三列是正交试验方案, 第四列为试验指标--收率
[1 1 1 31;
1 2 2 54;
1 3 3 38;
2 1 2 53;
2 2 3 49;
2 3 1 42;
3 1 3 57;
```

```

3 2 1 62;
3 3 2 64];
A = DATA(:, 1:NF); Y = DATA(:, NF+1);

for i=1:NL      % i:水平
    for j=1:NF  % j:因素 (列号)
        K(i, j) = sum(Y(find(A(:, j)==i)))
    end
end

k = K/r, R = max(k(:, :)) - min(k(:, :)) % R: 极差

```

(6) 结果分析

试验结果的极差分析

```

k = 41    47    45
      48    55    57
      61    48    48

```

最优生产条件：本例试验目的是提高收率，故应挑选每个因素（列） $k(:, j)$ 最大的那个水平。由矩阵 k 可知，在第一因素的第三水平、第二因素的第二水平和第三因素的第三水平下的试验条件，即温度 90°C 、压力 200kPa 、第 2 号催化剂，是比较好的试验条件。由于此条件在正交试验中没有出现过，因此需做验证试验。在此条件下进行试验，得到的产品收率为 74%，确实比正交试验中最好的第 9 次试验还要高。可以证明，当因素没有交互作用时，用此法选出的试验条件就是全面试验中最好的。因此，最优生产条件为 $T_3P_2C_2$ 。

用极差分析法确定各因素对指标影响的主次顺序：矩阵 k 的第 j 列中最大值与最小值之差，称为第 j 列（ j 因素）的极差。极差大，对试验指标影响大，为主要因素；极差小，对试验指标影响小，为次要因素。本例中，极差为 $R = [20 \ 8 \ 12]$ ，所以通过比较各因素的极差可知，各因素对指标影响的主次顺序为： $T \rightarrow C \rightarrow P$ （主 \rightarrow 次）。

8.3.2.2 有交互作用的正交实验

【例 8-2】乙酰胺苯磺化反应试验

试验目的：通过正交试验，寻找最佳操作条件以提高乙酰胺苯的产率。因素-水平见表 8-5。

表 8-5 因素-水平

因素 水平	反应温度 A/ $^{\circ}\text{C}$	反应时间 B/h	硫酸浓度 C/%	操作方法 D
1	50	1	17	搅拌
2	70	2	27	不搅拌

考虑到反应温度与反应时间、反应温度与硫酸浓度之间可能有交互作用，即考虑 $A \times B$ 、 $A \times C$ （交互作用以“ \times ”号表示，下同）。

(1) 根据自由度选取适当的正交表

用一张正交表设计试验时，表的行数恰好是试验点的个数，若不作重复试验，数据的总偏差平方和的自由度正好是这个表的行数减 1，故规定正交表的自由度 $f_{\text{总}} = \text{行数} - 1$ 。

考虑到 4 个因素 A、B、C、D 及交互作用 $A \times B$ 、 $A \times C$ ，总的自由度为 $f_{\text{总}} = 4 \times 1 + 2 \times 1 =$

6. 而正交表 $L_8(2^7)$ 的总自由度为 $8-1=7$ ，大于 $f_{\text{总}}$ ，故可选用 $L_8(2^7)$ 表来安排实验方案。

(2) 有交互作用时的表头设计

许多正交表都附有相应的交互作用表，利用它可以找出正交表中任意两列间的交互作用列。 $L_8(2^7)$ 的两列间的交互作用如表 8-6 所示。表中的数字均为列号，如要查第 1 列与第 2 列的交互作用列，则可从(1)横着往右看，从(2)竖着往上看，其交叉点为 3，故第 3 列就是第 1 列与第 2 列的交互作用列。如第 1 列排 A 因素，第 2 列排 B 因素，则第 3 列排它们的交互作用 A×B（可看作一个因素）。

表 8-6 $L_8(2^7)$ 的两列间的交互作用

1	2	3	4	5	6	7	列号
(1)	3	2	5	4	7	6	1
	(2)	1	6	7	4	5	2
		(3)	7	6	5	4	3
			(4)	1	2	3	4
				(5)	3	2	5
					(6)	1	6
						(7)	7

正交试验方案与结果如表 8-7 所示。

表 8-7 正交试验方案与结果

因 素 (列号)	A 反应温度	B 反应时间	A×B	C 硫酸浓度	A×C		D 操作方法	产率 Y/%
试验号	1	2	3	4	5	6	7	
1	1 (50℃)	1 (1 小时)	1	1 (17%)	1	1	1(搅拌)	65
2	1	1	1	2 (27%)	2	2	2(不搅拌)	74
3	1	2 (2 小时)	2	1	1	2	2	71
4	1	2	2	2	2	1	1	73
5	2 (70℃)	1	2	1	2	1	2	70
6	2	1	2	2	1	2	1	73
7	2	2	1	1	2	2	1	62
8	2	2	1	2	1	1	2	67
K	283 272	282 273	268 287	268 287	276 279		273 282	
k	70.75 68.00	70.50 68.25	67.00 71.75	67.00 71.75	69.00 69.75		68.25 70.50	
R	2.75	2.25	4.75	4.75	0.75		2.25	

有交互作用的正交试验结果的分析方法与无交互作用的情况完全相同，只是把交互作用列的相应K, k和极差R也算出，填入表中以便分析比较。所以，计算程序与OrthExpAnalysis1.m完全类似，只须修改NL、NF、r和DATA的数据，见OrthExpAnalysis2.m。

各因素及交互作用对试验指标影响的主次顺序：

从极差可以看出，各因素及交互作用对试验指标影响的主次顺序为：

A×B、C → A → B、D → A×C （主→次）

也就是说，A×B、C 是重要因素，A 是较重要因素，B、D 是次要因素，而 A×C 是次要因素，且它的极差明显小于其他因素的极差，因此可以忽略不计。

最优操作条件的确定：

对各因素水平的选取原则是：① 不涉及交互作用的因素（或交互作用可忽略的因素）

其最佳水平，仍然是平均值中指标最高的水平；② 有交互作用的因素，其水平的选取不能单独考虑，应画出二元表和相应的二元图，进行比较后再选择对指标较优的水平。

根据以上原则，由于 C、D 两因素不涉及交互作用（或忽略），可选其平均产率高的水平即 C₂D₂。而 A 与 B 间有交互作用，需画出二元表（表 8-8）。

表 8-8 A 与 B 之间的二元

B \ A	A	
	A ₁	A ₂
B ₁	$\frac{65+74}{2}=69.5$ (1, 2)	$\frac{70+73}{2}=71.5$ (5, 6)
B ₂	$\frac{71+73}{2}=72$ (3, 4)	$\frac{62+67}{2}=64.5$ (7, 8)

可见，A₁B₂和 A₂B₁ 平均产率都较高，且两者相差很小。从提高工效来看，用 A₂B₁（70℃，1 小时）比用 A₁B₂（50℃，2 小时）好，因为时间可减少一半，单位时间的产率较高。因此，最佳操作条件为：A₂B₁C₂D₂。

经试验验证，用此条件进行生产，产率确有提高。

8.3.3 正交实验结果的方差分析

前面介绍了正交试验设计的直观分析法，此法比较简便易懂，只要对试验结果作少量计算，通过综合比较，便可得出最优生产条件。但直观分析不能估计试验误差的大小，很难断定那个因素对试验结果影响显著，哪个因素影响不显著。为了弥补直观分析法的这个缺点，可采用方差分析的方法。方差分析正是将因素水平（或交互作用）的变化所引起的试验结果间的差异与误差的波动所引起的试验结果间的差异区分开来的一种数学方法。

8.3.3.1 无交互作用的正交实验

【例 8-3】 对[例 8-1]进行方差分析

正交试验方案与结果如表 8-9 所示。与例 8-1 中的表 8-3 不同的是，这里表 8-9 使误差（因素 E）占正交表的 1 列。

表 8-9 正交试验方案与结果

因素(列号)	A	B	C	E	指标
试验号	1	2	3	4	收率 Y/%
1	1 (80℃)	1 (150 kPa)	1 (1 号催化剂)	1	31
2	1	2 (200 kPa)	2 (2 号催化剂)	2	54
3	1	3 (250 kPa)	3 (3 号催化剂)	3	38
4	2 (85℃)	1	2	3	53
5	2	2	3	1	49
6	2	3	1	2	42
7	3 (90℃)	1	3	2	57
8	3	2	1	3	62
9	3	3	2	1	64
K	123	141	135	144	
	144	165	171	153	
	183	144	144	153	
Q	23118	22614	22734	22518	
S	618	114	234	18	

一般地, 设水平的数字为 $1, 2, \dots, s$, 每一个水平重复出现 $r = n/s$ 次。在正交表的每一列 j 中, 每一种水平 $(1, 2, \dots, s)$ 上的试验值 (指标) 之和分别为 $K_{1j}, K_{2j}, \dots, K_{sj}$ 。则 K 、 Q 和 S 的计算公式如下:

$$P = \frac{1}{n} \left(\sum_{i=1}^n Y_i \right)^2 \quad (8-18)$$

$$Q_j = \frac{1}{r} \sum_{i=1}^r K_{ij}^2 \quad (8-19)$$

$$S_j = Q_j - P \quad (S_j \text{ 为各因素的平方和}) \quad (8-20)$$

本例中, K 、 Q 和 S 的计算结果见表 8-9。

$$\text{总平方和:} \quad S_T = \sum_j S_j \quad (8-21)$$

$$\text{各因素的均方:} \quad \bar{S}_j = \frac{S_j}{f_j} \quad (j = 1, 2, \dots, m-1) \quad (8-22)$$

$$\bar{S}_E = \frac{S_E}{f_E} \quad (8-23)$$

$$\text{其中, 总自由度:} \quad f_T = n - 1 \quad (8-24)$$

$$\text{除了误差以外的其他因素的自由度:} \quad f_j = s - 1 \quad (8-25)$$

$$\text{误差的自由度:} \quad f_E = f_T - \sum f_j \quad (8-26)$$

$$F \text{ 比:} \quad F_j = \frac{\bar{S}_j}{\bar{S}_E} \quad (8-27)$$

$$\text{显著性检验: 若} \quad F_j = \frac{\bar{S}_j}{\bar{S}_E} > F_{1-\alpha}(f_j, f_E) \quad (8-28)$$

则因素 j 的作用显著。

本例的方差分析见表 8-10。

表 8-10 方差分析

方差来源	平方和	自由度	均方	F 值	F_{α} 值	显著性
A	618	2	309	34.33	$F_{0.05}(2,2)=19$	*
B	114	2	57	6.33	$F_{0.01}(2,2)=99$	Δ
C	234	2	117	13.00	$F_{0.1}(2,2)=9$	*
E	18	2	9		$F_{0.2}(2,2)=4$	
总和	984	8				

最优水平组合 (操作条件): 由表 8-9 的 K 值可知, 对于因素 A , 其三个水平 A_1 、 A_2 和 A_3 上的指标之和分别为 $K_{11} = 123$ 、 $K_{21} = 144$ 和 $K_{31} = 183$, 其中 K_{31} 最大, 故 A_3 是最优水平; 同样, 对于因素 B , B_2 是最优水平; 对于因素 C , C_2 是最优水平。所以最优水平组合为 (A_3, B_2, C_2) 。

程序清单 OrthExpAnalysis3.m

```
function OrthExpAnalysis3
```

```
% DATA: 正交试验方案及结果---最后一列为试验指标 (收率), 其余是以水平表示的正交
```

```
% 试验方案, NF: 因素数 (列数), NL: 水平数, r: 每个水平有 r 次试验
```

```
clear all; clc
```

```

DATA = ...           % 最后一列为试验指标（收率），其余是以水平表示的正交试验方案
    [1  1  1  1  31;
      1  2  2  2  54;
      1  3  3  3  38;
      2  1  2  3  53;
      2  2  3  1  49;
      2  3  1  2  42;
      3  1  3  2  57;
      3  2  1  3  62;
      3  3  2  1  64];
[m, n] = size(DATA);  NF = n-1;  NL = max(DATA(:, 1))
r = 0;
for i=1:m;
    if DATA(i, 1)==1
        r = r+1;      % 每个水平有 r 次试验
    end
end
A = DATA(:, 1:NF);  Y = DATA(:, NF+1);
% 计算每个因素（列）相同水平的指标总和 K
for i=1:NL
    % i:水平
    for j=1:NF
        % j:因素（列号）
        K(i, j) = sum(Y(find(A(:, j)==i)))
    end
end
end
xa = sum(Y)/length(Y),  xT = sum(Y),  P = 1/(NL*r)*xT^2
j=1:NF;  Q(j) = sum(K(:, j).^2)/r,  S = Q - P      % S: 各因素的平方和
ST = sum(S),  Sav = S/(NL-1)                      % ST: 总平方和, Sav: 各因素的均方
F = Sav(1:NF-1)/Sav(end)                          % F: 显著性检验

```

8.3.3.2 有交互作用的正交实验

【例 8-4】某一种抗菌素的发酵培养基由黄豆饼粉、蛋白胨、葡萄糖、碳源 1 号、 KH_2PO_4 、 CaCO_3 、无机盐 1 号等组成。现打算对其中五个成分的最适配比，以及最适装量，按三种水平进行试验，并将其两个成分（黄豆饼粉、蛋白胨）合并为一个因素，这样构成一个五因素三水平试验。需考虑的交互作用有 $A \times B$ 、 $A \times C$ 、 $A \times E$ 。因素-水平如表 8-11 所示。

表 8-11 因素-水平

因素 水平	黄豆粉+蛋白胨 A/%	葡萄糖 B/%	KH_2PO_4 C/%	碳源 1 号 D	装量 E/ml
1	1.0	4.5	0	0.5	30
2	2.0	6.5	0.01	1.5	60
3	3.0	8.5	0.03	2.5	90

表头设计:

根据因素（包括交互作用）及水平计算总自由度（诸因素与交互作用的自由度之和）： $\Sigma f_{\text{因}} = 5 \times (3-1) + 3(3-1)(3-1) = 22$ ，而正交表 $L_{27}(3^{13})$ 的自由度 $f_{\text{表}} = 27 - 1 = 26 > \Sigma f_{\text{因}}$ ，故可选 $L_{27}(3^{13})$ 正交表。见表 8-12。根据表头设计原则，表头设计如下：

因素	A	B	A×B	A×B	C	A×C	A×C	E	A×E	A×E	D
列号	1	2	3	4	5	6	7	8	9	10	11

表 8-12 试验方案与结果计算

试验号	因素(列号)													指标 Y/%
	A	B	A×B	A×B	C	A×C	A×C	E	A×E	A×E	D			
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.69
2	1	1	1	1	2	2	2	2	2	2	2	2	2	0.54
3	1	1	1	1	3	3	3	2	3	3	3	3	3	0.37
4	1	2	2	2	1	1	1	2	2	2	3	3	3	0.66
5	1	2	2	2	2	2	2	3	3	3	1	1	1	0.75
6	1	2	2	2	3	3	3	1	1	1	2	2	2	0.54
7	1	3	3	3	1	1	1	3	3	3	1	1	1	0.37
8	1	3	3	3	2	2	2	1	1	1	3	3	3	0.66
9	1	3	3	3	3	3	3	2	3	3	2	2	2	0.75
10	2	1	1	1	1	1	1	1	1	1	1	1	1	0.69
11	2	1	1	1	2	2	2	2	2	2	2	2	2	0.54
12	2	1	1	1	3	3	3	3	3	3	3	3	3	0.37
13	2	2	2	2	1	1	1	2	2	2	3	3	3	0.66
14	2	2	2	2	2	2	2	3	3	3	1	1	1	0.75
15	2	2	2	2	3	3	3	1	1	1	2	2	2	0.54
16	2	3	3	3	1	1	1	3	3	3	1	1	1	0.37
17	2	3	3	3	2	2	2	1	1	1	3	3	3	0.66
18	2	3	3	3	3	3	3	2	3	3	2	2	2	0.75
19	3	1	1	1	1	1	1	1	1	1	1	1	1	0.69
20	3	1	1	1	2	2	2	2	2	2	2	2	2	0.54
21	3	1	1	1	3	3	3	3	3	3	3	3	3	0.37
22	3	2	2	2	1	1	1	2	2	2	3	3	3	0.66
23	3	2	2	2	2	2	2	3	3	3	1	1	1	0.75
24	3	2	2	2	3	3	3	1	1	1	2	2	2	0.54
25	3	3	3	3	1	1	1	3	3	3	1	1	1	0.37
26	3	3	3	3	2	2	2	1	1	1	3	3	3	0.66
27	3	3	3	3	3	3	3	2	3	3	2	2	2	0.75
K	...													
Q	...													
S	...													

注：表 8-12 的正交表及实验结果参见光盘中的 OrthExpdata.mat，K、Q、S 可运行 OrthExpAnalysis4.m 得到。

方差分析

程序见光盘中的 OrthExpAnalysis4.m。运行该程序得到以下结果：

空白列（仅两列）的平方和之和 $S_{\text{empty}} = S(12) + S(13) = 0.064$ 。

交互作用的平方和为：

$$S_{AB} = S(3) + S(4) = 0.046 \quad \% \text{ AB: 第 3、4 列的平方和之和}$$

$$S_{AC} = S(6) + S(7) = 0.31 \quad \% \text{ AC: 第 6、7 列的平方和之和}$$

$$S_{AE} = S(9) + S(10) = 0.046 \quad \% \text{ AE: 第 9、10 列的平方和之和}$$

可见， S_{AB} 和 S_{AE} 均小于 S_{empty} ，交互作用 AB 和 AE 可不考虑，或者说，将第 3、4 列，第 9、10 列都看作空白列，并与第 12、13 的空白列一起并入误差中，即：

$$\text{误差 } e \text{ 的平方和为 } S_e = S_{\text{empty}} + S_{AB} + S_{AE} = 0.064 + 0.046 + 0.046 = 0.16。$$

这样，方差分析不再考虑交互作用 AB 和 AE。

方差分析的结果如表 8-13 所示。

表 8-13 方差分析

来 源	平 方 和	自 由 度	均 方	F 值	显 著 性
A	$S(1) = 0.90$	2	0.450	34.50	**
B	$S(2) = 0.05$	2	0.025	1.94	
C	$S(5) = 0.23$	2	0.115	8.86	*

续表

来 源	平 方 和	自 由 度	均 方	F 值	显 著 性
D	$S(11) = 0.07$	2	0.033	2.57	*
E	$S(8) = 0.10$	2	0.051	3.95	
AC	$S(6) \cdot S(7) = 0.31$	4	0.077	5.97	
误差 e	0.16	12	0.013		

查 F 分布表知, $F_{0.01}(2, 12) = 6.93$, $F_{0.01}(4, 12) = 5.41$, $F_{0.05}(2, 12) = 3.89$, $F_{0.05}(4, 12) = 3.26$, 所以显著的因素是 A、C 及其交互作用 AC。

最优操作条件的确定:

根据前面所述的各因素水平的选取原则, 由于 B、D 两因素不涉及交互作用 (或忽略), 可选其平均产率高的水平即 B_2D_2 。而 A 与 C 间有交互作用, 需画出二元表 (表 8-14)。

表 8-14 A 与 C 之间的二元关系

C \ A			
	A_1	A_2	A_3
C_1	$(0.69+0.66+0.81)/3 = 2.16/3$	$(0.93+0.86-0.99)/3 = 2.78/3$	$(0.69+0.86+0.66)/3 = 2.21/3$
C_2	$(0.54+0.75+0.68)/3 = 1.97/3$	$(1.15+0.97-1.13)/3 = 3.25/3$	$(1.10+1.16+1.38)/3 = 3.64/3$
C_3	$(0.37+0.48+0.39)/3 = 1.24/3$	$(0.90+1.17+0.80)/3 = 2.87/3$	$(0.91+1.30+0.73)/3 = 2.94/3$

可见, 因素 A 与 C 的水平组合应选 A_3C_2 。综上所述, 试验的最优水平组合为: $A_3B_2C_2D_2E_1$ 。但由于 B 和 D 都不是显著的因素, 所以最佳操作条件为 $A_3BC_2DE_1$, 其中 B 和 D 不指明水平, 意思是这两个因素不是显著因素, 其水平可任意选取 (一般根据实际选择较易实现的操作水平)。

8.4 序贯实验设计

序贯实验设计是一种很有效的方法, 它包括最佳序贯判别和最佳序贯估计两方面内容。

8.4.1 实验设计准则

(1) 最小容积设计—用于精确估计 (非线性) 机理模型中全部参数的设计

考虑一个具有 p 个待估计的参数 $\beta_1, \beta_2, \dots, \beta_p$ 及 q 个操作变量 (实验变量) $\xi_{1u}, \xi_{2u}, \dots, \xi_{qu}$ 的数学模型, 则对第 u 次试验:

$$\eta_u = f(\beta_1, \beta_2, \dots, \beta_p; \xi_{1u}, \xi_{2u}, \dots, \xi_{qu}), \quad u = 1, 2, \dots, n \quad (8-29)$$

n 为实验次数。

以向量形式表示为:

$$\eta_u = f(\beta, \xi_u) \quad (8-30)$$

其中, η_u 为第 u 次观测的响应 (真值), 如产物浓度、转化率等; β 为待估计的参数向量 (如反应速率常数和反应平衡常数), ξ_u 为变量 (如温度和压力)。

设计实验的目的是选择旨在得到最精确的 β 估计值的实验条件 (自变量格点)。

模型对第 i 个参数的偏导数为:

$$x_{iu} = \frac{\partial f(\beta, \xi_u)}{\partial \beta_i} \quad (i = 1, 2, \dots, p; u = 1, 2, \dots, n) \quad (8-31)$$

定义 $n \times p$ 阶矩阵 X :

$$X = \{x_{iu}\} = \begin{bmatrix} x_{11} & x_{21} & \dots & x_{p1} \\ x_{12} & x_{22} & \dots & x_{p2} \\ \dots & \dots & \dots & \dots \\ x_{1n} & x_{2n} & \dots & x_{pn} \end{bmatrix} \quad (8-32)$$

X 称为导数矩阵。由于模型（如动力学模型）一般对参数是非线性的，因此， X 将是待估计参数的函数。

为便于实验设计，将模型在当时的参数估计值 $\hat{\beta}$ 处作一级泰勒展开以逼近响应函数：

$$f(\beta, \xi_u) = f(\hat{\beta}, \xi_u) + X(\beta - \hat{\beta}) \quad (8-33)$$

采用此逼近方法，得到 $\hat{\beta}$ 的方差-协方差矩阵为：

$$V(\hat{\beta}) = (X'X)^{-1} \sigma^2 \quad (8-34)$$

其中 σ^2 为每个观测响应值的方差， X' 为 X 的转阵， $(X'X)^{-1}$ 为 $X'X$ 的逆阵。

对于线性模型，由上式可得到精确的参数方差和协方差；对于非线性模型，由上式可得到近似的参数方差和协方差。假定实验误差服从 $\mu = 0$ 的正态分布。Box 和 Lucas 的实验设计准则是使置信域的容积极小，故该准则称最小容积准则。由于 σ^2 未知，故设计准则只需考虑矩阵 $(X'X)^{-1}$ 。也就是说，该设计准则是选择操作变量的格点，使 $(X'X)^{-1}$ 的行列式极小，或使 $X'X$ 的行列式极大。这样选择实验点能有效地使 $\hat{\beta}$ 的总的方差测量值为最小，即置信域的容积达到极小。

最小容积设计方法的优点：所得置信域小，而且减少了被估参数之间的相关性，并可量度实验误差。

(2) 球形设计准则

球形设计准则就是选择操作变量的格点，使 $X'X$ 中最小的特征根为极大。这样设计的实验将导致椭球的最长的轴得到最大的收缩，即使置信区域的形状尽可能的圆。

值得注意的是，最小容积准则和形状准则的区别仅在于：前者对可操作域中的每一个格点计算 $\det X'X$ ，而后者计算 $X'X$ 的 p 个特征值。两者存在如下关系：

$$\det X'X = \prod_{i=1}^p \lambda_i \quad (8-35)$$

8.4.2 序贯实验设计步骤

用于精估模型参数的序贯实验设计步骤如下。

- ① 进行一些预实验（初始实验），这些实验条件必须包括实验域的大部分；
- ② 根据预实验结果，用非线性最小二乘法估计得到参数的初始估值；
- ③ 设计下一次实验：根据最小容积准则（或形状准则），利用最优化方法（Marquadt 法）

由实验数据及参数估值计算下次实验的自变量格点；

- ④ 完成这次实验，得到新的实验结果；
- ⑤ 将新实验结果与旧实验数据一起用来重新估算参数，得到新的参数估值；
- ⑥ 重复②~④反复进行设计-实验-参数估计等迭代过程，每设计和进行一次实验都最大

地降低置信区间的大小（或使置信区域的形状尽可能的圆），直到参数精度基本不能再提高为止。

8.5 MATLAB 实验设计函数

一般来说,即使严格的模型也有一些未知的常数,实验的目的就是获得能够用来估计这些常数的数据。实验设计的最大优点在于您主动地操纵所研究的系统。通过实验设计(Design of Experiments, DOE),您可以产生较少的实验点,但获得较高的信息质量。统计工具箱提供一些函数,用于产生适合于各种情况的实验设计:

- 完全析因设计(Full Factorial Designs): `fullfact()`、`ff2n()`
- 不完全析因设计(Fractional Factorial Designs): `fracfact()`
- 响应面设计(Response Surface Designs)
 - Box-Behnken 设计: `bbdesign()`
 - 中心区组设计(Central composite design): `ccdesign()`
- D-优化设计(D-Optimal Designs):
`cordexch()`、`rowexch()`、`daugment()`、`dcovary()`、`candexch()`和 `candgen()`。

下面主要介绍完全析因设计函数 `fullfact()`和 `ff2n()`、不完全析因设计 `fracfact()`、以及 D-优化设计的函数 `cordexch()`和 `rowexch()`等。

完全析因设计(Full Factorial Designs)

函数 `fullfact()`

功能: 进行完全析因设计

调用格式: `design = fullfact(levels)`

输入参数: `levels` 是一个向量,其各个元素分别指定 `design` 对应因素(列)的水平个数;
`design` 试验设定矩阵。

例如, `levels = [3 2]`时, `fullfact()`产生完全析因设计总共有 $3 \times 2 = 6$ 次试验,其中第一因素(列)有 3 个水平,第二因素(列)有 2 个水平。

执行命令

```
> design = fullfact( [3 2] )
```

得:

`design =`

1	1
2	1
3	1
1	2
2	2
3	2

函数 `ff2n()`

功能: 进行二水平完全析因设计

调用格式: `design = ff2n(n)`

输入参数 `n` 试验因素个数(`design`的列数):
`design` 试验设置,试验次数(`design`的行数)共有 2^n 次。

显然,函数 `ff2n()`是函数 `fullfact()`的特例,如 `ff2n(3)`等效于 `fullfact([2 2 2])`。

不完全析因设计 (Fractional Factorial Designs)

函数 fracfact()

功能：生成二水平的不完全析因设计

调用格式：`x = fracfact(gen)`

`[x, conf] = fracfact(gen)`

输入参数

gen 用于定义不完全析因设计的生成器字符串，它由空格隔开的一串“单词”所组成。每个单词描述输出设计列是如何由完全因子的列组成的。典型的情况是，**gen** 将为前面几个因子包含单字母单词，另外一些多字母单词描述剩下的因子是如何由前面几个因子所组成的。

输出参数

x 设计点的矩阵，它是一个二水平完全析因设计的一部分。假设在 **gen** 中有 **m** 个单词，且每个单词由字母表中的前 **n** 个字母的子集组成。输出矩阵 **x** 有 2^n 行、**m** 列。令 **F** 表示二水平完全析因设计，正如由函数 `ff2n(n)` 生成的一样。

conf 单元数组，它描述主效应的组成模式和所有的二因子交互作用。

【例 8-5】 通过试验研究 4 个因子对单一响应的影响效应，规定只能进行 8 次试验。试验目的是确定哪些因子对响应有影响。在有些因子配对之间可能会有交互作用。

进行完全析因设计需做 16 次试验，但如果假设没有三因子交互作用，则通过 8 次就可估计主因子效应。

```
>> [x, conf] = fracfact('a b c abc')
```

```
x =
```

-1	-1	-1	-1
-1	-1	1	1
-1	1	-1	1
-1	1	1	-1
1	-1	-1	1
1	-1	1	-1
1	1	-1	-1
1	1	1	1

```
conf =
```

'Term'	'Generator'	'Confounding'
'X1'	'a'	'X1'
'X2'	'b'	'X2'
'X3'	'c'	'X3'
'X4'	'abc'	'X4'
'X1*X2'	'ab'	'X1*X2 + X3*X4'
'X1*X3'	'ac'	'X1*X3 + X2*X4'
'X1*X4'	'bc'	'X1*X4 + X2*X3'
'X2*X3'	'bc'	'X1*X4 + X2*X3'

'X2*X4'	'ac'	'X1*X3 + X2*X4'
'X3*X4'	'ab'	'X1*X2 + X3*X4'

D-优化设计 (D-Optimal Designs)

从 20 世纪 70 年代开始, 统计学家们开始应用计算机优化技术进行试验设计。D-优化设计使 Fisher 信息矩阵 $X^T X$ 的行列式最大化。该矩阵与参数的协方差矩阵的逆成比例。所以, 使 $\det(X^T X)$ 最大化等价于使参数协方差矩阵的行列式最小化。D-优化设计使线性模型参数 β 的回归估计的置信椭球的体积最小化。

MATLAB 统计工具箱提供的 D-优化设计函数 `cordexch()` 和 `rowexch()` 用于计算给定模型的 D-优化设计, 两者都属于迭代算法, 它们通过改变元素的增量来改进初始设计。其中, 函数 `cordexch()` 采用坐标交换算法, 增量为设计矩阵的单个元素, 而函数 `rowexch()` 采用行交换算法, 增量为设计矩阵的行元素。下面介绍函数 `cordexch()` 和 `rowexch()` 的用法。

函数 `cordexch()`

功能: 用坐标交换算法来进行 D-优化试验设计。

调用格式: `[settings, x] = cordexch(nfactors, nruns, model)`

输入参数: <code>nfactors</code>	因素个数
<code>nruns</code>	希望进行的试验次数
<code>model</code>	(可选) 用于指定回归模型阶次的字符串, 其值如下:
	'linear' 或 'l' 默认模型, 带有常数项和线性项;
	'interaction' 或 'i' 包含常数项、线性项和交互项;
	'quadratic' 或 'q' 包含交互项和平方项;
	'purequadratic' 或 'p' 包含常数项、线性项和平方项。
输出参数: <code>settings</code>	因素设置矩阵, 共有 <code>nruns</code> 行、 <code>nfactors</code> 列;
<code>x</code>	有关设计矩阵。

函数 `rowexch()`

功能: 用行交换算法来进行 D-优化试验设计。该函数的用法与函数 `cordexch()` 相同。

【例 8-6】 考虑如下具有二输入变量的二次模型, 试用坐标交换算法进行 D-优化试验设计。

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_1^2 + \beta_5 x_2^2 + \varepsilon$$

式中, ε 表示误差。

解 该模型包含交互项和平方项, 共有两个因素和 6 个参数。

执行命令:

```
>> settings = cordexch(2, 8, 'q')
```

可得:

settings =

-1	1
1	1
-1	-1
0	0
-1	0
0	1

```
1    -1
0    -1
```

需注意的是, 由于模型共有 6 个待估计参数, 试验次数必须在 6 以上 (最低 6 次), 这里取 8 次, 可使估计的参数更准些。

【例 8-7】 考虑如下具有二输入变量的交互模型, 试用行交换算法进行 D-优化试验设计:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \varepsilon$$

式中, ε 表示误差。

解 该模型包含常数项、线性项和交互项, 共有两个因素(x_1, x_2)和 4 个参数($\beta_0, \beta_1, \beta_2, \beta_3$)。执行命令:

```
>> settings = rowexch(2, 5, 'i')
```

可得:

```
settings =
```

```
1    1
-1   1
-1  -1
1   -1
-1  -1
```

【例 8-8】 用于动力学参数估计的 D-最优试验设计

已知某反应系统的 Hougen-Watson 反应动力学模型为

$$y = \frac{\beta_1 x_2 - x_3 / \beta_3}{1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3} \quad (1)$$

式中, $\beta_1, \beta_2, \dots, \beta_5$ 是未知参数, 而 x_1, x_2 和 x_3 是 3 个输入变量, 分别表示氢气、戊烷和异戊烷的分压 p_1, p_2 和 p_3 , y 为反应速率。

已知实验操作范围是: $p_1 = 100 \sim 470$, $p_2 = 80 \sim 300$, $p_3 = 10 \sim 120$ 。

(a) 试用 D-优化设计方法进行最优实验设计, 并用非线性回归方法拟合实验数据, 以求取动力学参数。

(b) 采用尝试法改变操作条件做试验, 观察操作条件变化对反应速率的影响。并在允许的反应物分压范围内看看能否找到使反应速率达到最大的设置 (试验条件设置)。

解 (a) 用 D-优化设计方法

(i) 为便于模拟真实的实验情况, 先构造一个反应模拟器。

根据文献得到该反应系统的动力学参数为

$$\beta = [1.25, 1.5183, 0.064, 0.0378, 0.1326]$$

这样, 以此参数代入动力学模型 (1), 并引入误差, 即可模拟实验, 得到各种分压 $x_i = p_i$ 下的反应速率数据 y_i 。相应的 MATLAB 代码如下:

$$y = 1.25 * (p_2 - p_3 / 1.5183) / (1 + 0.064 * p_1 + 0.0378 * p_2 + 0.1326 * p_3) * \text{normrnd}(1, 0.02);$$

其中, $\text{normrnd}(1, 0.02)$ 产生均值 $\mu = 1$ 、方差为 $\sigma^2 = 0.02^2$ 的正态分布的随机数。

(ii) D-最优试验设计

用 MATLAB 坐标交换算法函数 `cordexch()` 来进行 D-优化试验设计 (安排 13 次实验):

```
settings = cordexch(3, 13, 'quadratic');
```

该命令产生 3×13 的 settings (因子设置) 矩阵, 即共有三个变量的 13 次试验。字符串 'quadratic'

指明按三个变量的完全二次模型 (full quadratic model) 拟合, 该模型为:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots \quad (\text{线性项})$$

$$b_{12}x_1x_2 + b_{13}x_1x_3 + b_{23}x_2x_3 + \dots \quad (\text{交互项})$$

$$b_{11}x_1^2 + b_{22}x_2^2 + b_{33}x_3^2 \quad (\text{二次项})$$

因子设置矩阵 **settings** 中的第 1、2、3 列向量分别代表因素 p_1 、 p_2 和 p_3 的水平设置, 各列中的 -1、0 和 1, 分别代表对应因素的三个水平中的低水平、中水平和高水平。

考虑三个因素 (氢气、戊烷和异戊烷的分压 p_1 、 p_2 和 p_3) 的 3 个实验水平如下:

settings 中的元素	p_1	p_2	p_3
1 (高)	470	300	120
0 (中)	285	190	65
-1 (低)	100	80	10

把因子设置矩阵 **settings** 变换为相应的实验条件矩阵 **expCond**。

(iii) 根据反应模拟器计算对应实验条件 **expCond** 的反应速率 y , 即得到实验数据 **data** = [**expCond**, y]。

(iv) 进行数据分析: 采用 Gauss-Newton 法最小二乘拟合函数 **nlinfit()**, 以此 **data** 实验数据来估计模型 (1) 中的参数。也可使用 **nlintool()** 拟合。

该程序为 **DOE.m**, 它同时使用最小二乘拟合函数 **nlinfit()** 和 **nlintool()**。运行该程序将发现, 每执行一次程序得到的 **settings/expCond** 矩阵不一样, 但每次都能找到使反应速率达到最大的设置 (试验条件设置) ($p_1 = 100$, $p_2 = 300$, $p_3 = 10$)。

(b) 采用尝试法进行试验设计

由于共有三个因素三个水平, 完全析因实验需做 $3^3 = 27$ 次实验。显然, 采用尝试法改变条件做 13 次实验, 除非运气特别好以外很难找到使反应速率达到最大的试验设置 ($p_1 = 100$, $p_2 = 300$, $p_3 = 10$), 这部分作为读者练习。

程序清单 **DOE.m**

```
function DOE
clear all; clc
% 各因素的水平
p = [470, 300, 120
      285, 190, 65
      100, 80, 10];
% D-优化试验设计: 先生成 settings, 再将 settings 转变为相应实验条件 expCond
settings = cordexch(3, 13, 'q'); % settings: 因子设置矩阵
mr = p(2, :); mr = mr(ones(13, 1), :); % mr: middle row, i.e., middle level
hr = (p(1, :) - p(3, :))/2; hr = hr(ones(13, 1), :);
expCond = settings.*hr + mr; % expCond: settings 的相应实验条件
% 由反应模拟器生成实验数据 data

p1 = expCond(:, 1); p2 = expCond(:, 2); p3 = expCond(:, 3);
y = zeros(13, 1);
for k = 1:13
```

```

y(k) = 1.25*(p2(k) - p3(k)/1.5183)./(1 + 0.064*p1(k) ...
      + 0.0378*p2(k) + 0.1326*p3(k))*normrnd(1, 0.02);
enddata = [expCond y] % data 为实验数据矩阵(实验条件及其对应的反应速率)
% 数据分析: 由非线性模型 Nonlinear Model 估计参数

x = data(:, 1:3); y = data(:, 4);
xname = str2mat('Hydrogen', 'n-Pentane', 'Isopentane'); yname = 'Reaction Rate';
beta0 = [1.2 0.1 0.01 0.1 1.5]; % 参数初值
nlintool(x, y, @hougen, beta0, [], xname, yname);
rstool(x, y, 'quadratic', [], xname, yname);
[beta, resid, j] = nlinfit(x, y, @hougen, beta0)
ci = nlparci(beta, resid, j)
% 参数辨识结果:  $\beta_1$ 、 $\beta_2$ 、...  $\beta_5$ 

fprintf('Estimated Parameters:\n')
fprintf('\t $\beta_1 = %.2f \pm %.2f$ \n', beta(1), ci(1, 2) - beta(1))
fprintf('\t $\beta_2 = %.3f \pm %.3f$ \n', beta(2), ci(2, 2) - beta(2))
fprintf('\t $\beta_3 = %.4f \pm %.4f$ \n', beta(3), ci(3, 2) - beta(3))
fprintf('\t $\beta_4 = %.4f \pm %.4f$ \n', beta(4), ci(4, 2) - beta(4))
fprintf('\t $\beta_5 = %.4f \pm %.4f$ \n', beta(5), ci(5, 2) - beta(5))

```

8.6 化工数据处理指南

在化工数据处理方面,本章已叙述了正交实验数据的极差分析和方差分析,其他方面的数据处理已分散于其他章节中,现整理如下,以方便读者阅读:

- 绘制二维曲线图和三维曲面图(第2章2.9);
- 实验数据插值(第3章3.1.1);
- 实验数据(曲线)平滑和最小二乘拟合(第3章3.1.2);
- 数值积分和数值微分(第3章3.2);
- 实验数据曲线拟合、参数估计/回归分析(第7章);
- 极差分析和方差分析(第8章);
- 神经网络函数逼近(第9章)。

参 考 文 献

- 1 郑明东,刘炼杰,余亮,姚伯元编著.化工数据建模与试验优化设计.第一版.合肥:中国科学技术大学出版社,2001.6. 17
- 2 Alkis Constantinides & Navid Mostoufi. Numerical Methods for Chemical Engineers with MATLAB Applications. Prentice Hall, 1999
- 3 上海市科学技术交流站.正交试验设计法—多因素的试验方法.上海:上海人民出版社,1979
- 4 胡上序,陈德钊编著.实验数据的分析和处理.杭州:浙江大学出版社,1996
- 5 何为,陈际达编.优化试验设计法及其在化学中的应用.成都:电子科技大学出版社,1994
- 6 化学工业部化工科研计算机应用中心站译.序贯实验设计方法译文集.北京:化学工业出版社,1983
- 7 Atkinson, A.C. and A.N. Donev. Optimum Experimental Designs. Oxford Science Publications, 1992
- 8 Douglas M. Bates and Donald G. Watts, Nonlinear Regression Analysis and Its Applications. John Wiley & Sons. 1988, 271~272 中译本:韦博成,万方焕,朱宏图,张绕庭.非线性回归分析及其应用.第一版.北京:中国统计出版社,1997, 11
- 9 Box, G. E. P., W.G. Hunter, and J.S. Hunter. Statistics for Experimenters. Wiley, New York. 1978

- 10 Montgomery, D.C., Design and Analysis of Experiments, Wiley, New York, 1984. 中译本: [美]Douglas C. Montgomery. 实验设计与分析 (第三版). 北京: 中国统计出版社, 1998
- 11 John H. Seinfeld and Leon Lapidus. Mathematical Methods in Chemical Engineering Volume 3 Process Modeling, Estimation, and Identification. New Jersey: Prentice-Hall Inc., 1974
- 12 Englezos & Kalogerakis. Applied Parameter Estimation for Chemical Engineers. Marcel Dekker, 2000 (16.1.3)
- 13 Statistics Toolbox User's Guide, Version 4.0, The MathWorks, Inc., 2002

第9章 神经网络在化工中的应用

9.1 概述

人脑是宇宙中已知最复杂、最完善和最有效的高级信息处理系统。人工神经网络(Artificial Neural Network, 简称 ANN)是在人类对其大脑神经网络认识理解的基础上人工构造的能够实现某种功能的神经网络。它是理论化的人脑神经网络的数学模型, 是基于模仿大脑神经网络结构和功能而建立的一种信息处理系统。

人工神经网络(简称神经网络, NN)是由许多并行互联的相同神经元模型组成的, 网络的信号处理由神经元之间的相互作用来实现。神经网络实际上描述了网络输入和输出之间的一种函数关系。通过选取不同的模型结构和传递函数, 可以形成各种不同的神经网络, 得到不同的输入-输出关系式, 并达到不同的设计目的, 完成不同的任务。

常规数学模型化与神经网络模型化的简单比较

回顾第7章, 数学模型(通常指机理模型或经验模型)中未知参数的估计(辨识)方法如图9-1所示, 这是常规的数学模型化方法。已知实验数据集(输入 i , 目标 i), 实验序号 $i=1, 2, \dots, N$, 并给定参数初值, 则模拟输出与目标之间的比较, 可以两者之间的残差平方和最小为收敛准则。因此, 利用最优化方法可估计出未知参数。这样, 参数估计后的数学模型能够预测不同输入变量下的输出值。

从大体的计算过程看, 神经网络模型化与常规数学模型化比较相似, 都采用最优化方法来确定未知参数, 如图9-2所示。在这里, 神经网络代替了常规数学模型, 而权值和阈值(bias, 也称偏差)就是待估计的未知参数。所谓“调节权值和阈值”实际上也是利用最优化方法, 通过输出与目标之间的比较加以调整, 直至网络输出与目标相匹配为止, 从而估计出权值和阈值, 这一过程称为网络训练。这样, 网络训练好(或权值和阈值调节好)后的神经网络, 也能够预测不同输入变量下的输出值。

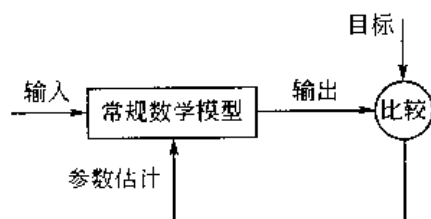


图 9-1 常规模型参数估计

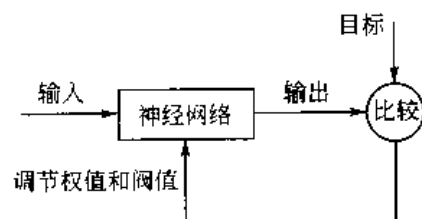


图 9-2 神经网络训练

神经网络模型化与常规数学模型化也有明显的不同。一般来说, 常规数学模型化以较少(通常小于10)的变量和一个响应为基础。如果要求来自同一变量集的更多响应, 则使用常规数学模型化方法时, 通常必须为每一个响应分别建立一个独立模型。而神经网络都能进行单响应和多响应的模型化。此外, 常规数学模型具有明显的物理意义, 大多利用“三传一反”原理导出, 但只适用于特定的物理过程, 而神经网络由许多并行互联的神经元组成的输入输出关系模型, 虽然缺乏内在的物理意义, 但通用性很广, 只要有足够的实验或工业数据集,

通常都可以用来预测不同输入变量下的结果。

神经网络的化工应用

神经网络在化学工程和生化工程中的应用主要有以下四类。① 故障诊断和特征分类；② 预测和优化；③ 时变系统的过程预测、模型化和控制；④ 混合使用神经网络和专家系统对复杂过程的预测与设计。

神经网络的有关化工应用研究很多，例如化工过程模型预测、优化和控制；化工系统的动态模拟；发酵过程的神经网络预测和控制；化工设备故障诊断；软测量；化工产品配方研究；催化剂配方研究；在化学分析方面的应用：蛋白质结构分析、谱分析（如红外光谱、质谱）；环保废水处理，等等。

在模型化方面，由于神经网络具有并行性和高度非线性等特点，因此它常常作为一种黑箱模型，用来模拟难于用机理模型建立的复杂过程。

下面几节首先介绍神经元模型和网络结构，这是神经网络的理论与应用基础，接着介绍用于过程预测和模型化方面的神经网络，包括线性神经网络、BP神经网络和径向基神经网络等，重点结合示例介绍基于 MATLAB 神经网络工具箱的函数调用方法。然后介绍利用神经网络工具箱 nntool GUI（图形用户界面）的求解方法。最后给出化工应用实例。需要说明的是，神经网络计算需要提供足够多的数据集，这比较花费时间，由于时间关系，本章暂时未能提供足够多的化工应用实例，但仿照书中所介绍的简单示例，更复杂的化工应用问题应该能够解决。不过，作者打算在下一版本中尽量增加有关应用实例，以更方便读者学习使用。

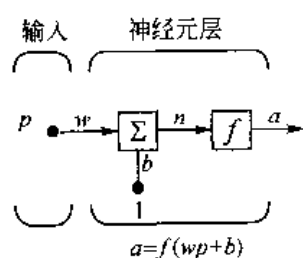


图 9-3 简单神经元

9.2 神经元模型与网络结构

9.2.1 神经元模型

9.2.1.1 简单神经元

一个简单的神经元如图 9-3 所示，它是一个单输入单输出系统。图中 p 为标量输入， w 为标量权值， b 为阈值（阈很像一个权，只是其输入恒定为 1）， f 为传递函数（ f 是通用符号，具体的传递函数将在后面介绍），变量 $n = wp + b$ 作为 f 的输入。其网络输出 a 为：

$$a = f(n) = f(wp + b)$$

其中， w 和 b 是两个可调节的标量参数。神经网络的主要思想就是通过调节参数 w 和 b 对网络进行训练，以达到所希望的目标（如网络计算输出值与实验结果之间的残差平方和最小）。

9.2.1.2 有一向量输入的神经元

具有一个向量输入的神经元如图 9-4(a)所示，其中输入列向量 \mathbf{p} 有 R 个元素： p_1 、 p_2 、...、 p_R ，权值矩阵 \mathbf{W} 为 $w_{1,1}$ 、 $w_{1,2}$ 、...、 $w_{1,R}$ ， $n = \mathbf{W}\mathbf{p} + b$ 作为传递函数 f 的输入（或自变量）。这是最基本的一个神经元模型，其网络输出 a 为：

$$a = f(\mathbf{W}\mathbf{p} + b)$$

式中， $\mathbf{W}\mathbf{p} = w_{1,1}p_1 + w_{1,2}p_2 + \dots + w_{1,R}p_R$ 。因这里只是一个神经元，故 \mathbf{W} 为单行矩阵。

图 9-4(b)是与 9-2(a)等效的简图，图中的黑方框表示输入向量 \mathbf{p} （列向量），向量和矩阵的大小在其变量名的下方，如 \mathbf{W} 的大小为 $1 \times R$ 。

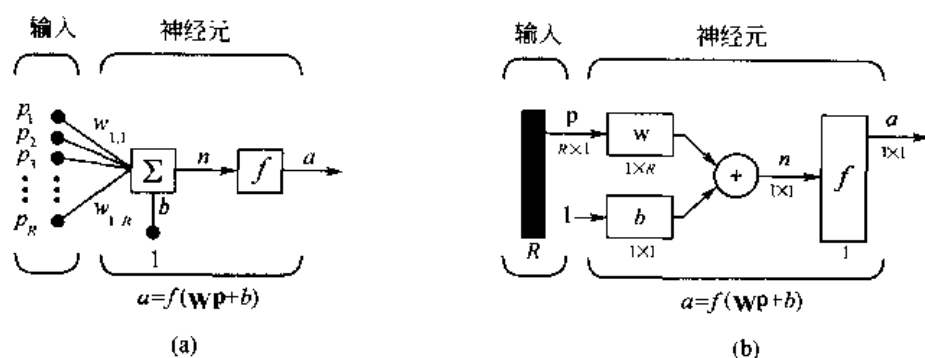


图 9-4 有一向量输入的神经元

9.2.1.3 传递函数

传递函数是神经元的核心，它反映神经元的输入输出关系。MATLAB 神经网络工具箱提供许多传递函数，下面仅介绍将要用到的几个常用传递函数：纯线性函数 `purelin()`、对数 sigmoid 型函数 `logsig()`、双曲正切 sigmoid 型函数 `tansig()` 和径向基函数 `radbas()`。

(1) 纯线性函数 `purelin()` 其输入输出函数关系为

$$f(n) = n$$

其中 n 为网络输入（下同），如图 9-5(a)所示。

MATLAB 程序调用方法： `a = purelin(n)`

(2) 对数 sigmoid 型函数 `logsig()` 其输入输出函数关系为

$$f(n) = 1 / (1 + \exp(-n))$$

如图 9-5(b)所示。

调用方法： `a = logsig(n)`

(3) 双曲正切 sigmoid 型函数 `tansig()` 其输入输出函数关系为

$$f(n) = 2 / (1 + \exp(-2 * n)) - 1$$

如图 9-5(c)所示。

调用方法： `a = tansig(n)`

(4) 径向基函数 `radbas()` 其输入输出函数关系为

$$f(n) = \exp(-n^2)$$

如图 9-5(d)所示，调用方法： `a = radbas(n)`

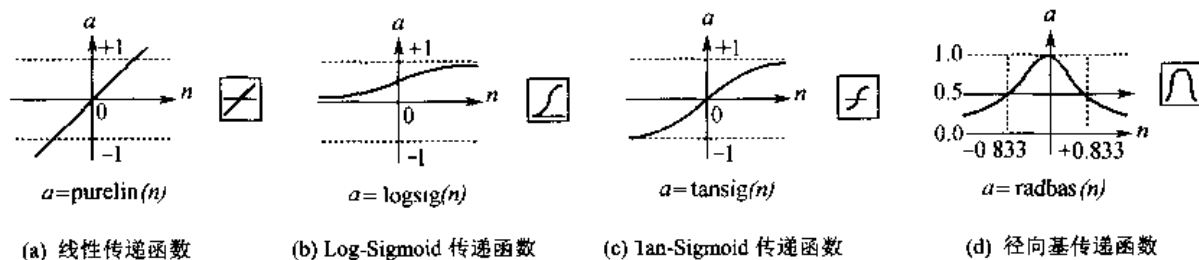


图 9-5 传递函数

由图可直观看出有关传递函数的特征（输入和输出值范围、形状或变化规律等）。由图 9-5(b)、(c)可见，对数 Sigmoid 型函数和双曲正切 Sigmoid 型函数均是 S 形状的曲线，故这两个函数统称为 S 型(Sigmoid)函数。以上各图的右侧图标是对应传递函数的简图，将在神经网络结构图中用来形象表示所用的传递函数。

传递函数 `purelin()` 可用于线性近似 (参见 9.3), `logsig()` 和 `tansig()` 常用于 BP 神经网络 (参见 9.4), 而 `radbas()` 用于径向基神经网络 (参见 9.5)。

9.2.2 网络结构

9.2.2.1 单层神经网络

只有一层神经元的单层网络如图 9-6 所示, 其中图(b)是图(a)等效的简图。图中 \mathbf{p} 是长度为 R 的输入列向量 (同前)。与图 9-4 不同的是, 图 9-6 的神经元层包含 S 个神经元, 而不是仅仅一个神经元。因此, \mathbf{W} 是 $S \times R$ 矩阵, \mathbf{b} 、 \mathbf{n} 和 \mathbf{a} 都是长度为 S 的列向量。

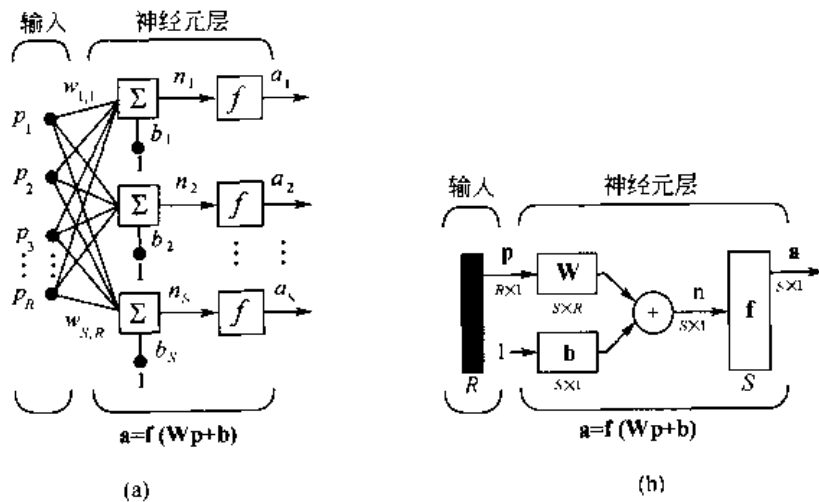


图 9-6 单层神经网络

网络输出为: $\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$

式中, \mathbf{a} 、 \mathbf{W} 、 \mathbf{p} 和 \mathbf{b} 如下:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \cdots & \cdots & \cdots & \cdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix}$$

由图 9-6(b)和网络输出表达式 $\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$ 可以看出, 神经元层由权矩阵 \mathbf{W} 、乘法运算符、阈向量 \mathbf{b} 、加和器和传递函数方框等组成。

9.2.2.2 多层神经网络

对于多层神经网络, 必须区分与输入相连的权矩阵和连接层与层之间的权矩阵, 并识别权矩阵的源(source)和目标(destination)。我们把与输入相连的权称为输入权(input weights), 而把层与层之间的连接权称为层权(layer weights), 相应的这两种权矩阵分别称为输入权矩阵和层权矩阵。此外, 用上标来识别各层网络不同的权、阈和传递函数等项的源(第 2 个 index)和目标(第 1 个 index)。例如, 若某个多层神经网络的第 1 层结构与图 9-6 中的单层神经元一样, 则此第 1 层网络结构可由图 9-6(b)经稍加修改后得到, 如图 9-7 所示。图中, 输入权矩阵(与输入向量 \mathbf{p} 连接)标记为 $\mathbf{IW}^{1,1}$, 它连接源 1(第 2 个 index)和目标 1(第 1 个 index)。第 1 层网络的阈向量 \mathbf{b}^1 、传递函数输入向量 \mathbf{n}^1 、输出向量 \mathbf{a}^1 、神经元数 S^1 等项, 均以 1 为其上标, 用来表示是属于第 1 层的变量。

图 9-7 中 $IW^{1,1}$ 与对应的 MATLAB 代码如下:

$IW^{1,1} \rightarrow \text{net.IW}\{1,1\}$

因此, 以下代码可获得传递函数的网络输入:

$n\{1\} = \text{net.IW}\{1,1\} * p + \text{net.b}\{1\};$

对于多层神经网络, 第 2 层以后还存在层权矩阵 (LW)。

以三层神经网络为例, 完整的网络结构如图 9-8(a) 和 9-8(b)所示, 其中 9-8(b)是 9-8(a)的简图。此三层神经网络的第 1 层有 R 个输入和 S^1 个神经元。由于每个中间层的输出就是其相邻下一层的输入。因此, 第 2 层的输入向量为 a^1 , 输出向量为 a^2 , 它有 S^1 个输入、 S^2 个神经元和一个 $S^2 \times S^1$ 权矩阵 $LW^{2,1}$ 。由于第 2 层的所有向量和矩阵都已确定, 因此它本身可以看成是一个单层神经网络, 其他网络层与此类似。

通常把最后产生网络输出的那层称为输出层 (以 y 表示), 而所有其他层称为隐含层。

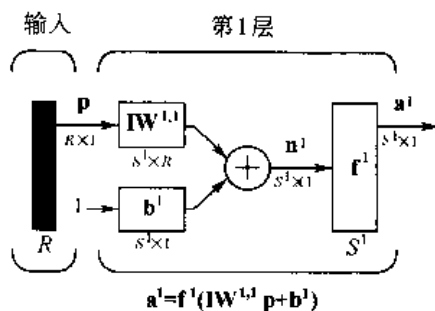


图 9-7 多层神经网络中的第 1 层

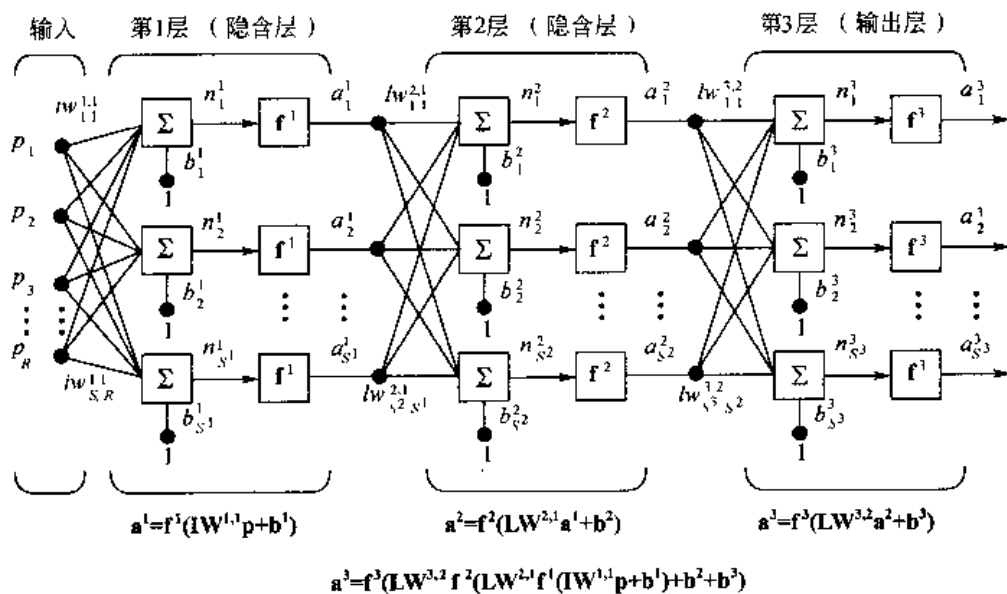


图 9-8(a) 三层神经网络

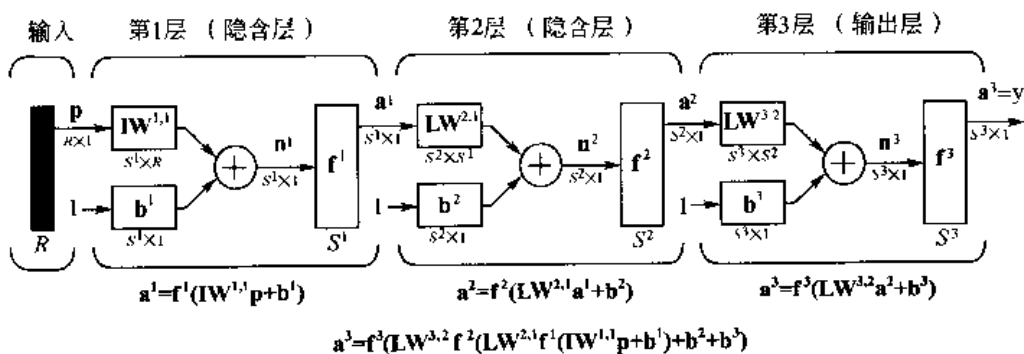


图 9-8(b) 三层神经网络简图

由于多层网络功能很强大，因而应用比较广泛。以二层网络为例，第1层用 sigmoid 型传递函数，第2层用线性传递函数，经过训练后能够很好的逼近任意函数。这种二层网络结构广泛应用于反向传播（BP）网络中（参见 9.4）。

9.3 线性神经网络

线性神经网络是最简单的一种神经网络，它由一个或多个线性神经元组成。由于每个神经元的传递函数为线性函数，线性神经网络的输出可以取任意值，因此，线性神经网络常用作多层神经网络的最后一层。线性神经网络可采用 Widrow-Hoff 学习规则或者 LMS（Least Mean Square）算法来调整网络的权值和阀值。

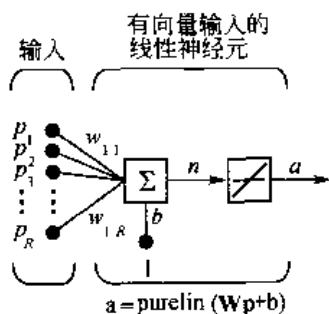


图 9-9 线性神经元

9.3.1 线性神经元模型

图 9-9 描述了一个有 R 个输入的线性神经元。该线性神经元使用一个线性传递函数 `purelin()`，其输入输出关系之间是简单的比例关系。因此，单个神经元的输出可表示为

$$a = \text{purelin}(\mathbf{W} \cdot \mathbf{p} + b) = \mathbf{W} \cdot \mathbf{p} + b$$

网络输出也可以用仿真函数 `sim()` 来求： $a = \text{sim}(p, W, b)$

9.3.2 线性神经网络结构

典型的单层线性神经网络如图 9-10 所示（图（b）是图（a）的简化表示形式），它有 R 个输入，具有 S 个神经元的网络层通过权值矩阵 \mathbf{W} 与输入 \mathbf{p} 相连。这种网络也称为 Madaline 网络。

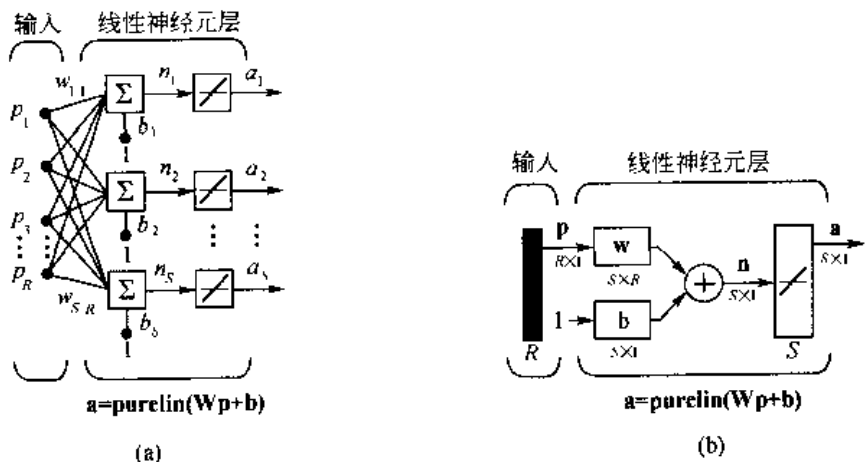


图 9-10 线性神经网络

9.3.3 线性神经网络的 MATLAB 求解方法

主要函数

函数 `newlin()`：用于创建、初始化线性神经网络；

函数 `sim()`：用于仿真。

函数 `newlin()` 的使用方法

`net = newlin`

% 以对话框方式创建一个新的线性网络

`net = newlin(PR, S, ID, LR)`

输入参数: PR 由 R 个输入的最小元素和最大元素组成的 R 行 2 列矩阵
 S 输出向量中元素的个数
 ID 输入延迟向量, 默认 = [0].
 LR 学习速率, 默认 = 0.01;
 输出参数: net 新的线性网络层

例如, 执行以下 MATLAB 代码:

```
net = newlin([-1 1; -1 1], 1);
net.IW{1, 1} = [2 3];
net.b{1} = [-4];
p = [5; 6];
a = sim(net, p)
```

则结果显示为: $a = 24$ 。

线性神经网络的初始化、训练及参数调整

通用函数

(1) 函数 `init()` 对神经网络初始化时, 将权值和阈值取为很小的正数或负数。

`net = init(net)`

(2) 函数 `train()` 训练一个神经网络

调用格式: `net = train(net, P, T)`

`[net, tr, Y, E, Pf, Af] = train(net, P, T, Pi, Ai, VV, TV)`

输入参数:

net 初始的神经网络.
 P 网络输入 (input signal)
 T 网络目标 (target signal) (可选), 默认值 = 0
 Pi 初始输入延时条件 (可选), 默认值 = 0
 Ai 初始网络层延时条件, 默认值 = 0
 VV 验证向量的结构, 默认值 = [], 训练信号变量有单元数组和矩阵两种格式
 TV 检验向量的结构, 默认值 = [], 训练信号变量有单元数组和矩阵两种格式

输出参数:

net 新的网络
 tr 训练记录 (epoch 和 perf)
 Y 网络输出 (the output signal)
 E 网络误差 (the error signal)
 Pf 最终输入延时条件 (可选)
 Af 最终网络层延时条件

(3) 函数 `adapt()` 使网络自适应地调整网络权值和阈值 (滤波信号)。

调用格式: `[net, Y, E] = adapt(net, P, T)`

`[net, Y, E, Pf, Af, tr] = adapt(net, P, T, Pi, Ai)`

变量同函数 `train()`。

线性神经网络的设计

与其他网络结构不同, 如果输入/目标向量已知, 则线性网络可以直接设计。利用函数

`newlind()`可以直接计算得到详细的网络权值和网络阈值，其调用格式为：

```
net = newlind % 以对话框方式创建一个新的线性网络
```

```
net = newlind(P, T, Pi)
```

例如，假设输入向量和目标向量为：

```
p = [1 2 3];
```

```
t = [2.0 4.1 5.9];
```

则下面语句可以用来设计一个线性网络：

```
net = newlind(p, t);
```

```
y = sim(net, p) % y = [2.05 4.00 5.95]
```

线性神经网络的学习规则

线性神经网络采用 Widrow-Hoff 学习规则，可利用函数 `learn` 求得权值和阈值的修正值。

学习的语句为：

```
e = t - a;
```

```
[dw, db] = learnwh(p, e, lr);
```

式中，`lr` 是学习率。当 `lr` 较大时，学习过程加快；但是当 `lr` 过大时，学习过程将变得不稳定，且误差将会加大，因此学习率 `lr` 的取值是至关重要的。用函数 `maxlinlr()` 可以为线性网络求得合适的学习率，该函数用法如下：

```
lr = maxlinlr(P)
```

```
lr = maxlinlr(P, 'bias')
```

它们分别对应于没有阈值和有阈值时的最大学习率。一旦学习率确定，线性神经网络可以利用 `learnwh()` 进行训练：

```
a = sim(net, x);
```

```
e = t - a;
```

```
[dw, db] = learnwh(p, e, lr);
```

线性神经网络的训练

`learnwh()` 可完成线性神经网络的训练，步骤是：（1）调用 `sim()` 进行仿真计算；（2）计算网络误差 `e`；（3）用 `learnwh()` 求得网络新的权值和阈值；（4）直到网络误差达到要求为止。

线性自适应网络

图 9-11 表示一个自适应网络，图中 TDL 是 Tapped Delay Line 的缩写。网络可通过 `adapt()` 来训练：

```
[net, Y, E, Pf, Af, tr] = adapt(net, P, T, Pi, Ai)
```

【例 9-1】 已知输入样本向量为 $p = [1.0, -1.2]$ ，目标向量为 $t = [0.5, 1.0]$ ，试设计一个单层线性神经网络。

程序说明 用 `newlind()` 设计一个单层线性神经网络使误差最小，并用 `sim()` 模拟计算网络。程序包括：

（1）定义输入向量和目标向量；（2）用 `errsurf()` 在可能的权值和阈值范围内计算网络误差，并用

`plotes()` 绘制误差曲面；（3）设计网络：用 `newlind()` 设计一网络使误差最小；（4）检验网络

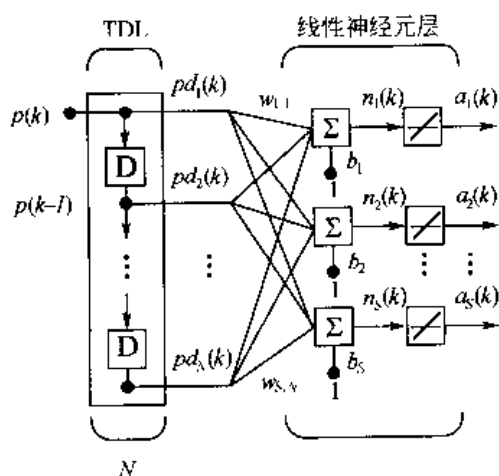


图 9-11 自适应网络

性能。

程序清单 xNewlind.m

```
function xNewlind
clear all; clc
% (1)定义输入向量和目标向量
p = [1.0 -1.2];          % p defines two 1-element input patterns (column vectors);
t = [0.5 1.0];          % t defines the associated 1-element targets (column vectors);
% (2)用 errsurf()在可能的权值和阈值范围内计算网络误差,并用 plotes()绘制误差曲面(error
surface)
w_range = -1:0.1:1;  b_range = -1:0.1:1;
es = errsurf(p, t, w_range, b_range, @purelin);
plotes(w_range, b_range, es);    % 以等值线绘制误差曲面,最佳的权值和阈值使曲面中出
现最低点
% (3)用 newlind()设计一网络使误差最小
net = newlind(p, t);
a = sim(net, p);          % 对输入 p 模拟计算网络
sse = sumsq(r(t - a));    % 神经元的残差平方和
plotep(net.IW{1, 1}, net.b{1}, sse); % plotep()用 newlind()返回的权值和阈值绘制网络的“位置”
% (4)检验(测试)输入为-1.2时,是否得到正确的目标值:1
p = -1.2;  a = sim(net, p)
```

计算结果 检验结果 $a = 1$, 正确。图形从略。

【例 9-2】 利用线性网络进行线性预测,即用 newlind()设计线性网络,在已知五个过去值的情况下,预测将来值。已知信号 t 的特征为:持续 5s, 采样频率为 40 次/秒。则信号 t 为:

```
time = 0:0.025:5;
t = sin(time*4*pi);
```

取信号 t 的最后 5 个值作为网络输入 p 。

程序说明 用 newlind()设计一个线性网络,并用 sim()对此线性网络进行仿真。该网络利用过去五个信号值可以对下一个信号进行预测。因网络有 5 个输入和 1 个输出(信号预测值),故应当采用具有 5 个输入的单神经元网络结构。实际上,函数 newlind()能够根据其输入参数 p 和 t 自动识别网络结构。

程序清单 xLinPred.m

```
clear all; clc
% (1) 定义一个波形(输入向量和目标向量)
%-----
time = 0:0.025:5;  % time: 定义模拟的时间步长
t = sin(time*4*pi); % t: 定义要预测的信号
q = length(t);
```

```

% 把信号 t 的最后 5 个值作为网络输入向量 p:
p = zeros(5, q); p(1, 2:q) = t(1, 1:(q-1)); p(2, 3:q) = t(1, 1:(q-2));
p(3, 4:q) = t(1, 1:(q-3)); p(4, 5:q) = t(1, 1:(q-4)); p(5, 6:q) = t(1, 1:(q-5));
% (2) 绘制要预测的信号图
plot(time, t), xlabel('时间, s'); ylabel('目标信号'); title('待预测信号');
% (3) 用 newlind() 设计网络并求取权值和阈值, 使线性神经元能够模拟系统
net = newlind(p, t);
% (4) 测试预测器——用 sim() 模拟线性神经元以预测在每个时间步长的下一个信号值
a = sim(net, p);
% (5) 绘制输出信号和目标信号
plot(time, a, time, t, '+')
xlabel('时间, s'); ylabel('信号'); legend('输出', '目标'); title('输出与目标信号');
% 绘制神经元输出信号与目标信号之差, 以便观察预测性能
figure, e = t - a; plot(time, e), hold on, plot([min(time) max(time)], [0 0], 'r'), hold off
xlabel('时间, s'); ylabel('误差'); title('误差信号');

```

计算结果 线性网络的预测性能如图 9-12 所示。由图可见, 输出信号与目标信号一致; 从图 9-12 (b) 可看出, 在初始阶段误差较大, 但随后误差很快趋近于 0。

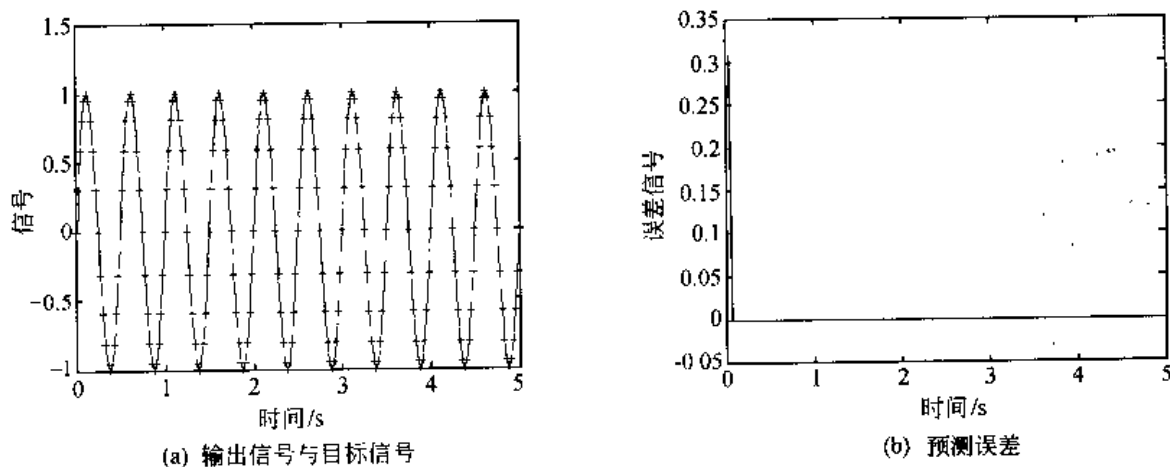


图 9-12 线性网络的预测性能

总结 [例 9-1] 和 [例 9-2] 可知, 有关神经网络的程序一般包括以下几方面内容:

- ① 定义输入向量和目标向量;
- ② 用 `plot()` 绘制要预测的信号, 或用 `errsurf()` 计算在可能的权值和阈值范围内的网络误差, 并用 `plotes()` 绘制误差曲面;
- ③ 设计网络;
- ④ 检验或测试网络性能, 并绘制性能图形;
- ⑤ 用训练好的网络来预测或模拟计算。

【例 9-3】 利用线性网络进行自适应预测。

利用 `adapt()` 对线性网络进行自适应训练, 在线修正权值和阈值, 这样网络就可以对信号

在过去和将来时刻的值的及时作出反应，从而对时变信号序列进行预测。已知信号 t 的特征为：持续 6 秒，前 4 秒每秒采样 20 次，从第 4 秒开始，频率突然加倍，这样 4 至 6 秒的频率是 0 至 4 秒的频率的两倍。则信号 t 为：

```
time1 = 0:0.05:4;
time2 = 4.05:0.024:6;
time = [time1 time2];
T = con2seq([sin(time1*4*pi) sin(time2*8*pi)]);
取信号  $t$  的最后 5 个值作为网络输入  $p$ 。
```

程序说明 利用线性网络进行自适应预测(Adaptive linear prediction)：用 `newlin()` 创建、初始化一个线性网络，并调用 `adapt()`，即用 Widrow-Hoff 学习规则训练此线性网络。这样，使用过去五个信号值，线性神经元就能够进行自适应训练，从而对下一个信号进行预测。

程序清单 *xAdapLinPred.m*

```
clear all; clc
% 定义一个波形
%-----
time1 = 0:0.05:4;      % time1 定义从 0~4 秒的时间段
time2 = 4.05:0.024:6;  % time2 定义从 4~6 秒的时间段
time = [time1 time2];  % time: 定义模拟的所有时间段 (0~6 秒) 的步长
t = con2seq([sin(time1*4*pi) sin(time2*8*pi)]); % t: 定义一个曾经改变频率的信号
p = t; % 网络输入与目标相同，网络将使用过去五个目标值来预测下一个目标值
% 绘制要预测的信号图
% -----
plot(time, cat(2, t{:})), xlabel('时间, s'); ylabel('目标信号'); title('待预测信号');
% 定义 (设计) 网络
% -----
% 设置学习率为 0.1 及五个延时信号，然后用 newlin() 创建一个线性网络，
% 所得的网络将用过去五个目标值来预测下一个目标值
lr = 0.1; delays = [1 2 3 4 5];
net = newlin(minmax(cat(2, p{:})), 1, delays, lr);
[net, y, e] = adapt(net, p, t); % 自适应地滤波信号
% 绘制网络输出信号
% -----
figure, plot(time, cat(2, y{:}), 'b-', time, cat(2, t{:}), 'k--') % 绘制线性神经元信号和目标信号
xlabel('时间, s'); ylabel('信号'); legend('输出', '目标'); title('输出与目标信号');
% 绘制神经元输出信号与目标信号之差 (网络误差)，以便观察预测性能
figure, plot(time, cat(2, e{:}), [min(time) max(time)], [0 0], 'r')
xlabel('时间, s'); ylabel('误差'); title('误差信号');
```

计算结果 自适应线性网络的预测性能如图 9-13 所示。

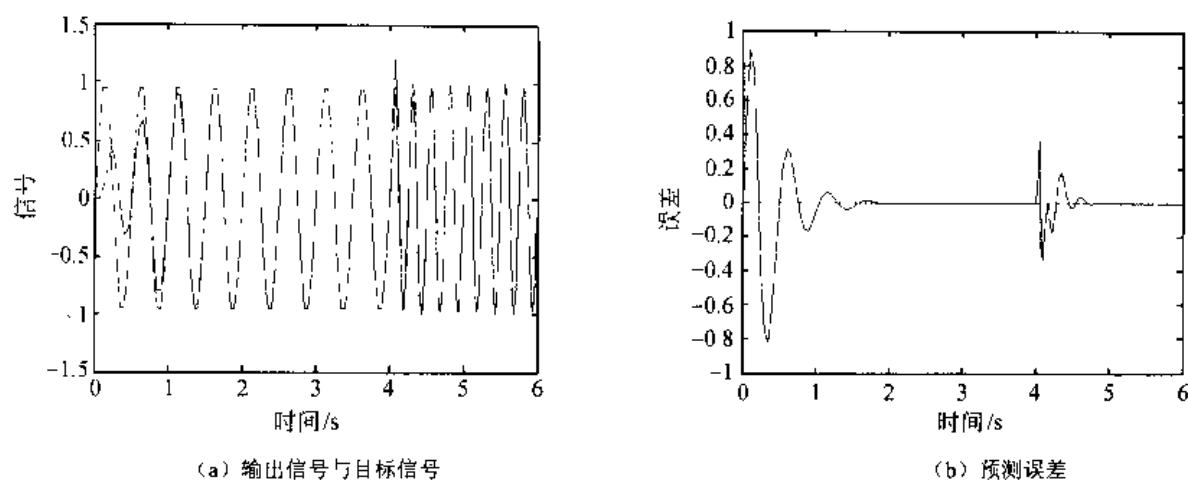


图 9-13 自适应线性网络的预测性能

9.4 BP 神经网络

本节介绍如何用 NN 工具箱中的 BP（反向传播）训练函数来训练前向神经网络（feedforward neural networks），以解决具体问题。训练过程一般包括四个步骤：（1）汇集数据；（2）创建网络对象；（3）训练网络；（4）仿真网络对新输入的响应。

9.4.1 BP 神经元模型

一个基本的 BP 神经元模型与图 9-14 相同，它具有 R 个输入，每个输入都通过一个适当的权值 w 与下一层相连，网络输出可表示为 $a = f(Wp + b)$ 。

在 BP 多层网络中，隐层神经元的传递函数（transfer function）通常使用 log-sigmoid 型函数 `logsig()`，也可以使用 tan-sigmoid 型函数 `tansig()`，有些场合还可以用纯线性函数 `purelin()`。这三个传递函数已在 9.2.1 中叙述。

如果 BP 多层网络的最后一层是 sigmoid 型神经元，那么网络的输出将限制在一个很小的范围内。如果最后一层是 `purelin` 型线性神经元，那么网络的输出可以取任意值。

这三个函数是 BP 网络最常用的传递函数，不过用户也可以根据需要自定义传递函数。

9.4.2 BP 网络结构

典型的单层网络结构如图 9-14 所示（图（b）是图（a）的简化图），它有 S 个 `logsig` 神经元，每个神经元有 R 个输入。

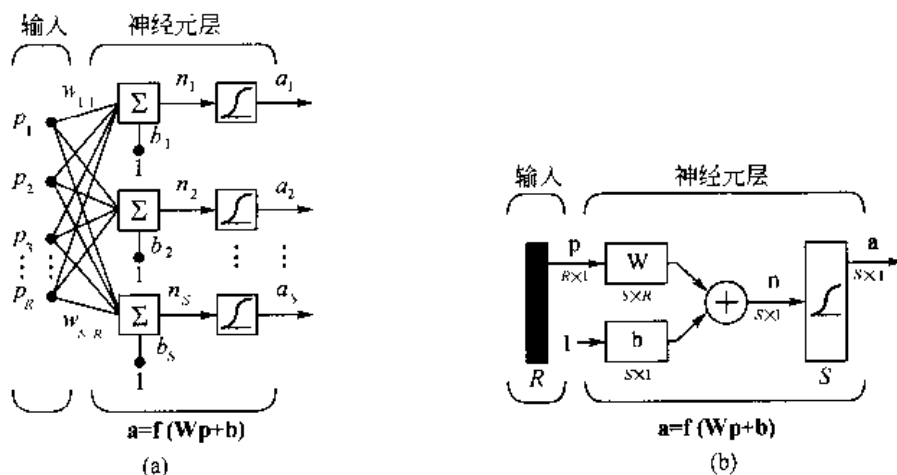


图 9-14 BP 网络结构

前向网络通常有一个或多个隐层，隐层神经元采用 Sigmoid 型传递函数，输出层神经元则采用线性传递函数。具有一个隐层的 BP 网络如图 9-15 所示。由于隐层神经元采用非线性传递函数 tansig ，故该网络可用来逼近非线性函数。

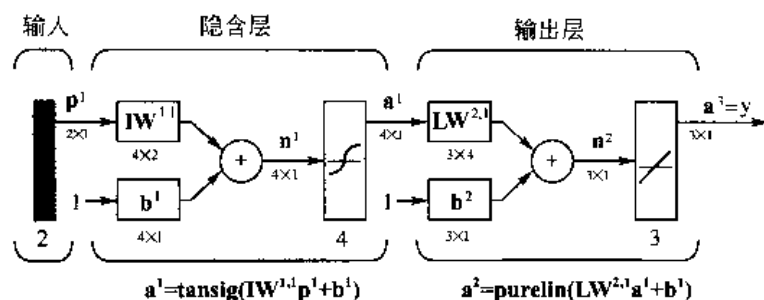


图 9-15 具有一个隐层的 BP 网络

9.4.3 BP 网络的设计

函数 `newff()` 用于创建一个前向 BP 网络 (feed-forward backpropagation network)，其调用格式为：

```
net = newff    % 用对话框创建一个新的前向 BP 网络
net = newff(PR, [S1 S2...SN], {TF1 TF2...TFN1}, BTF, BLF, PF)
```

输入参数：

- PR R 个输入向量的元素最小值和最大值构成的 $R \times 2$ 矩阵。
- Si 第 i 层的大小，共有 $N1$ 层。
- TFi 第 i 层的传递函数，可以是 `tansig()`、`logsig()` 或 `purelin()` 的一种，默认 'tansig'。
- BTF 反向传播网络训练函数，可以是 `trainlm()`、`trainbfg()`、`trainrp()`、`traingd()` 等其中的一种函数，默认为 'trainlm'。
- BLF 反向传播权值/阈值学习函数，可用 `learngd()` 或 `learngdm()`，默认为 'learngdm'。
- PF 性能函数，可以是 `mse()` 或 `msereg()`，默认为 'mse'。

返回结果：

net N 层前馈反向传播网络。

注意：`trainlm()` 的速度很快，但它需要许多内存空间。如果训练时出现 "out-of-memory" 错误，可尝试以下其中一种方法：

- ① 通过设置 `net.trainParam.mem_reduc = 2` 或大于 2 的值，以减慢 `trainlm()` 的训练速度(详见帮助功能：`>help trainlm`)；
- ② 使用 `trainbfg()`，该函数速度较慢，但其内存使用效率高于 `trainlm()`；
- ③ 使用 `trainrp()`，该函数速度较慢，但其内存使用效率高于 `trainbfg()`。

【例 9-4】 应用两层 BP 网络来完成函数逼近的任务，其中隐层神经元个数取 10 个。

程序说明 程序用 `newff()` 创建两层前向网络，并用 `sim()` 对网络进行仿真。

程序清单 *xNewff.m*

```
function xNewff
clear all; clc
% 输入样本 p 和目标 t
```

```

p = -1:1:1;
t = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609 ...
     .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988 ...
     .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
% 用 newff() 创建两层前向网络, 网络输入范围是[-1 1], 第一层有 10 个
% tansig 神经元, 第二层有 1 个 PURELIN 神经元。训练函数为 trainlm()
net = newff([-1 1], [10 1], {'tansig' 'purelin'}, 'trainlm');
% 对前向网络进行仿真, 并将网络输出值对目标值作图
y = sim(net, p)
plot(p, t, 'o', p, y, 'x'), title('训练向量 (训练 1 个 epoch)');
xlabel('输入向量 p'); ylabel('目标向量 t');
% Here the network is trained for 50 epochs. Again the network's output is plotted.
net.trainParam.epochs = 50; net = train(net, p, t); y = sim(net, p)
plot(p, t, 'o', p, y, '*-'), title('训练向量 (训练 50 个 epoch)');
xlabel('输入向量 p'); ylabel('目标向量 t');

```

计算结果 见图 9-16。

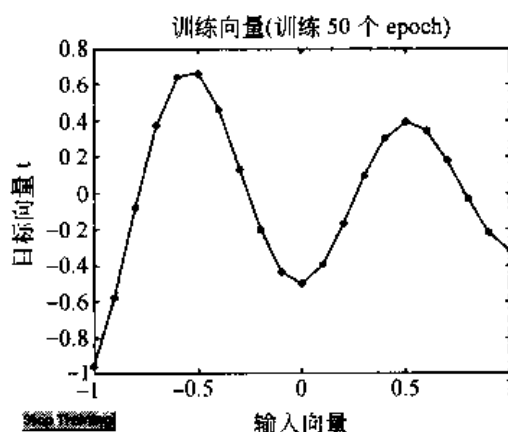


图 9-16 函数逼近结果

9.5 径向基神经网络

径向基神经网络 (Radial Basis Networks, RBN) 可能比标准的前向网络 (feed-forward-backpropagation networks) 需要更多的神经元, 但其所需的训练时间只相当于 FFBN 的一小部分。当有许多训练向量时, RBN 工作得很好。详细可参考 Chen (1991)^[3]。径向基神经网络有两个变种网络: 普遍化回归网络 (Generalized Regression Networks, GRNN) 和或然神经网络 (Probabilistic Neural Networks, PNN), 前者可用于函数逼近, 后者可用于分类, 这两种神经网络可参考 Wasserman (1993)^[4]。下面介绍 RBN 和 GRNN 网络。

9.5.1 径向基神经网络的重要函数

径向基神经网络的设计函数: newrb()、newrbe()。

GRNN 的设计函数: newgrnn()。

在命令窗口执行命令“help radbasis”即可查看所有的函数列表。

9.5.2 径向基神经元模型

图 9-17 表示一个有 R 个输入的径向基神经网络。径向基函数神经元的传递函数为 $\text{radbas}()$ (高斯函数)，其输出为 radbas 的输出， radbas 的输入为输入向量 \mathbf{p} 与权值向量 \mathbf{w} 的距离乘以阈值 b ，这与其他网络不同。图中 $\|\text{dist}\|$ 接受输入向量 \mathbf{p} 和权值矩阵 \mathbf{w} 的单一 1 行，并使两者之间进行点乘运算。径向基神经元的传递函数为 $\text{radbas}(n) = e^{-n^2}$ ，请参见图 9-5(d)。

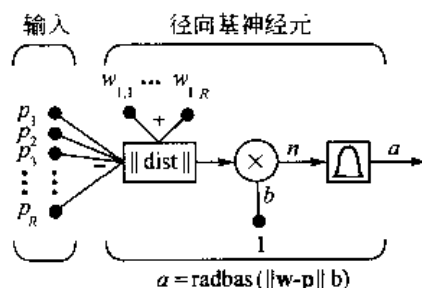


图 9-17 有 R 个输入的径向基函数神经元

9.5.3 径向基函数网络的结构

径向基函数网络的结构如图 9-18 所示，它包括两层：含 S^1 个神经元的径向基隐含层和含 S^2 个神经元的线性输出层。其中， R 为输入向量的元素个数， S^1 和 S^2 分别是第 1、2 层的神经元个数， a_i^1 是 \mathbf{a}^1 的第 i 个元素， $\mathbf{IW}^{1,1}$ 是由 $\mathbf{IW}^{1,1}$ 的第 i 行生成的向量。 $\|\text{dist}\|$ 框接受输入向量 \mathbf{p} 和输入权矩阵 $\mathbf{IW}^{1,1}$ ，产生一个具有 S^1 个元素的向量，其元素是输入向量 \mathbf{p} 与 $\mathbf{IW}^{1,1}$ 之间的距离。 $\|\text{dist}\|$ 的计算结果与阈向量 \mathbf{b}^1 之间进行点乘（“ \cdot ”，即对应元素相乘）操作，即得到 \mathbf{n}^1 。

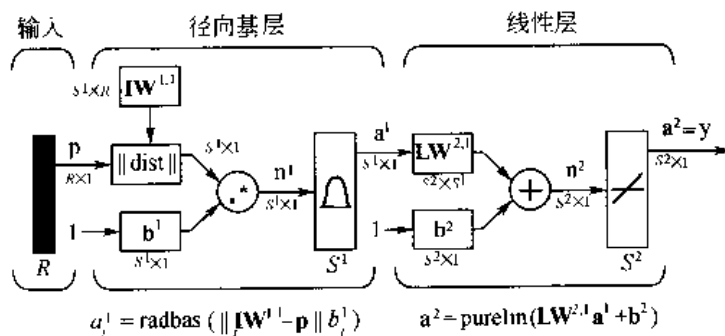


图 9-18 径向基函数网络的结构

前向神经网络 net 的第 1 层输出可用下列代码计算：

```
a{1} = radbas(netprod(dist(net.IW{1,1}, p), net.b{1}))
```

实际上，我们不必用此烦琐代码来计算，因为设计函数 $\text{newrbe}()$ 和 $\text{newrb}()$ 已包含网络设计的细节，网络层输出可用 $\text{sim}()$ 计算得到。简言之，每层网络的输出 \mathbf{a}^1 和 \mathbf{a}^2 可直接由设计函数 $\text{newrb}()$ 或 $\text{newbce}()$ 及函数 $\text{sim}()$ 得到，如：

```
net = newrb(P, T);
```

```
y = sim(net, x)
```

9.5.4 径向基网络的设计

函数 $\text{newrbe}()$ 用于设计径向基网络，可以达到零误差，其调用格式为：

```
net = newrbe % 以对话框方式创建一个新的径向基神经网络（零误差）
```

```
net = newrbe(P, T); % 输入 P: R x Q 矩阵，目标 T: S x Q 矩阵
```

```
y = sim(net, x) % Here the network is simulated for a new input.
```

设计的径向基网络可以精确输出目标值。

函数 newrb() 用于设计一个径向基神经网络，调用格式为：

```
net = newrb % 以对话框方式创建一个新的径向基神经网络
[net, tr] = newrb(P, T, goal, spread, MN, DF)
```

输入参数：

P 包含 Q 个输入向量的 $R \times Q$ 矩阵。
T 包含 Q 个目标向量的 $S \times Q$ 矩阵。
goal 残差平方均值指标(mean squared error goal)，默认值 = 0.0。
spread 径向基函数的分布常数 spread，默认值 = 1.0。
MN 最大的神经元数，默认为 Q 。
DF 默认值为 25。

输出参数：

net 返回一个新的径向基神经网络。

9.5.5 普遍化回归神经网络 (GRNN) 的设计

GRNN 的网络结构 (图 9-19) 与 RBN 的网络结构很类似，只是第 2 个网络层稍有不同。

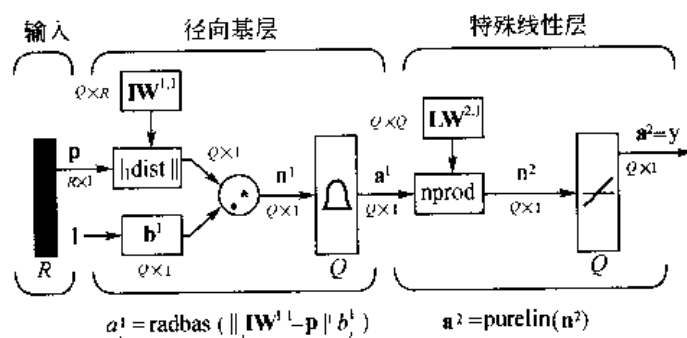


图 9-19 普遍化回归神经网络的结构

图 9-19 中， a_i^1 是 a^1 的第 i 个元素； $iW^{1,1}$ 是由 $IW^{1,1}$ 的第 i 行元素组成的一个向量。 $nprod$ 框 (相应函数为 normprod) 产生具有 S^2 个元素的向量 n^2 ，每个元素实际上是 $LW^{2,1}$ 的一行与输入向量 a^1 之间的点乘结果，且每个元素均通过 a^1 的所有元素加和进行归一化。例如，已知：

$LW\{1,2\} = [1 \ -2; 3 \ 4; 5 \ 6];$

$a\{1\} = [7; -8];$

则

$aout = \text{normprod}(LW\{1,2\}, a\{1\})$

$aout =$

-23

11

13

GRNN 的设计函数 newgrnn()：

```
net = newgrnn % 以对话框方式创建一个 GRNN
```

```
net = newgrnn(P, T, spread); % 用 newgrnn() 创建一个普遍化回归神经网络，参数同上。
```

9.5.6 示例

【例 9-5】 应用径向基网络来完成函数逼近任务

程序说明 程序用 newrb()创建径向基网络,并用 sim()对网络进行仿真。

程序清单 xNewrb.m

```
function xNewrb
clear all; clc
% 输入样本 p 和目标 t
P = -1:1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609 ...
      .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988 ...
      .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
% 用 newrb()创建径向基网络以逼近由 p 和 t 定义的函数
eg = 0.02;          % 残差平方和指标(sum-squared error goal)
sc = 1;             % 径向基函数的分布常数(spread constant)
net = newrb(P, T, eg, sc);
x = -1:0.01:1; y = sim(net, x); % 用 sim()对网络进行仿真
% 绘制图形以观察网络性能(重画训练集 p、t, 并绘制模拟曲线)
plot(p, t, '^', x, y); title('训练向量'); xlabel('输入向量 p'); ylabel('目标向量 t');
```

计算结果 用径向基网络实现函数逼近的结果如图 9-20 所示。

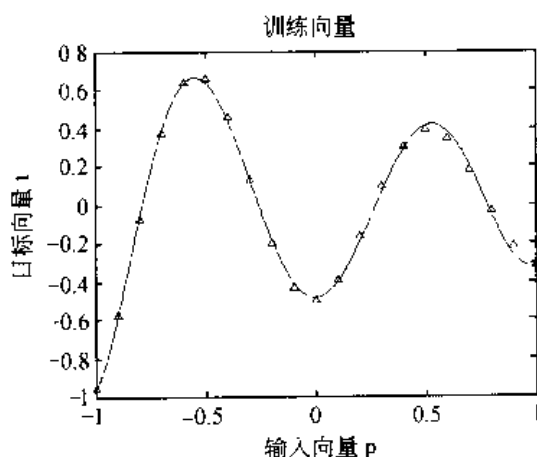


图 9-20 径向基网络的函数逼近

【例 9-6】 已知输入向量 $p = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8]$ 和目标向量 $t = [0 \ 1 \ 2 \ 3 \ 2 \ 1 \ 2 \ 1]$, 用普遍化回归神经网络 (GRNN) 实现函数逼近。

程序说明 用函数 newgrnn()创建普遍化回归神经网络,并用 sim()进行仿真。

程序清单 xNewgrnn.m

```
function xNewgrnn
clear all; clc
p = [1 2 3 4 5 6 7 8]; % inputs p
t = [0 1 2 3 2 1 2 1]; % target outputs t
spread = 0.7;          % a smaller spread would fit data better but be less smooth.
```

```

net = newgrnn(p, t, spread);% 用 newgrnn()设计一个普遍化回归神经网络
a = sim(net, p);
% 模拟计算网络对多个输入值的响应
cla reset
p2 = 0:1:9; a2 = sim(net, p2);
plot(p2, a2, 'linewidth', 3, 'color', [1 0 0]), hold on, plot(p, t, '.', 'markersize', 20)
axis([0 9 -1 4]), title('函数逼近'), xlabel('p 和 p2'), ylabel('t 和 a2')

```

计算结果 用普遍化回归神经网络实现函数逼近的结果如图 9-21 所示。

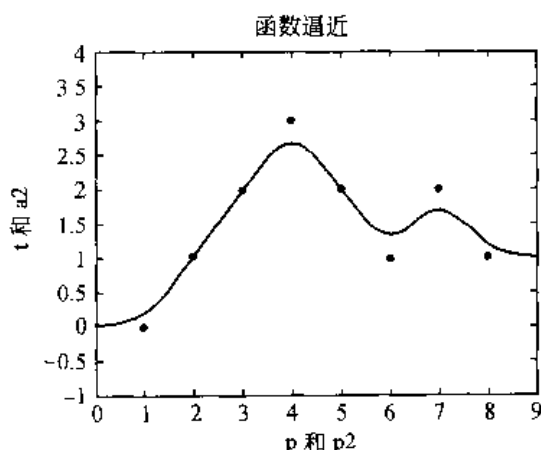


图 9-21 普遍化回归神经网络的函数逼近

9.6 神经网络工具箱 nntool GUI 求解方法

与 PDE 工具箱类似，MATLAB 神经网络工具箱不仅提供命令行函数的编程方式，还提供强大的 nntool—GUI（图形用户界面）求解方式。

在 MATLAB 工作环境中执行命令“>nntool”，即可打开 nntool 的网络/数据管理窗口。MATLAB 的 nntool 图形用户界面主要功能如下：

- 建立神经网络；
- 将数据输入至 GUI 中；
- 对网络进行初始化、训练和模拟（仿真）；
- 将训练结果从 GUI 输出至命令行工作环境中；
- 将命令行工作环境中的数据输入到 GUI 中。

对于一个新的问题，使用 nntool 能够：

(a) 用[NEW DATA]按钮创建新的输入数据和目标数据，或用[IMPORT]按钮从 MATLAB 工作环境中或一个文件中输入数据；

(b) 用[NEW NETWORK]按钮创建新的网络，或用[IMPORT]按钮从 MATLAB 工作环境中或一个文件中输入一个网络；

(c) 在网络列表中选择网络，再单击[TRAIN...]按钮以打开用来训练网络的窗口；

(d) 使用网络窗口来训练网络、模拟网络或执行其他像初始化权值或编辑权值这样的任务。

实际上，MATLAB 神经网络工具箱的 nntool 求解思路，与命令行函数编程求解思路是完

全一样的，因此，只要掌握命令行函数编程求解方法，就很容易掌握 nntool 的求解方法。

下面结合实例详细介绍 nntool 的求解方法。

【例 9-7】 用 nntool 求解[例 9-4]的相同问题：应用两层 BP 网络来完成函数逼近的任务，其中隐层神经元个数取 10 个。

解 由于使用 nntool 的求解思路和步骤与用神经网络工具箱函数通过编程的求解思路和步骤完全一样，为此这里用 nntool 求解时，完全按[例 9-4]的程序 xNewff.m 的相同内容及顺序逐步求解，具体步骤如下。

(1) 运行 nntool 后，弹出如图 9-22 所示的网络/数据管理器。

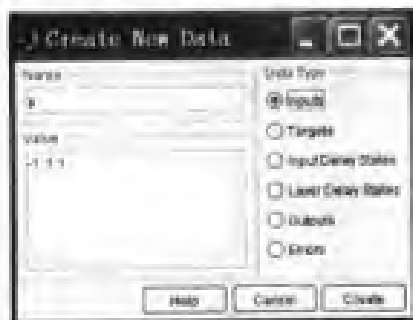
(2) 把下面已知的输入向量 p 和目标向量 t 输入到网络/数据管理器中：

```
p = -1:1:1;
t = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609 ...
     .1336 -.2013 -.4344 -.5000 -.3930 -.1647 .0988 ...
     .3072 .3960 .3449 .1816 -.0312 -.2189 -.3201];
```

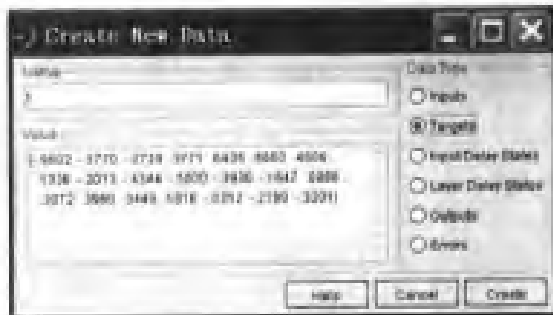
具体方法是，在网络/数据管理器界面上按[New Data...]，即弹出如图 9-23(a)所示的对话框，在 Name 的文本框中输入变量名： p ，在 Value 文本框中输入该变量的值： $-1:1:1$ （也支持 Ctrl+C 和 Ctrl+V 的通用拷贝操作），再选中 Data Type 的单选按钮 Input，并用鼠标点击[Create]，即完成输入向量 $p = -1:1:1$ 的输入工作。类似地，也可以输入目标向量 t ，不同之处仅在于这里选中 Data Type 的单选按钮 Target，如图 9-23(b)所示。



图 9-22 网络/数据管理器



(a)



(b)

图 9-23 创建新数据对话框

(3) 创建两层前向网络。例 9-4 的程序 xNewff.m 是用下列语句实现这一步骤的：

```
net = newff([-1 1], [10 1], {'tansig' 'purelin'}, 'trainlm');
```

用 nntool 实现这一步骤的方法是，在网络/数据管理器主界面上点击[New Network...]，出现如图 9-24 的 Creat New Network 对话框，然后进行与上述命令语句等效的设置。

① Input ranges: $[-1 \ 1]$ ，可直接键入 $[-1 \ 1]$ ，或在该文本框的右侧下拉条中选择“Get from input:p”后， $[-1 \ 1]$ 会自动填入文本框中。

② 设置 Properties for layer1 的 Number of neurons（神经元个数）：10，取 Transfer Function

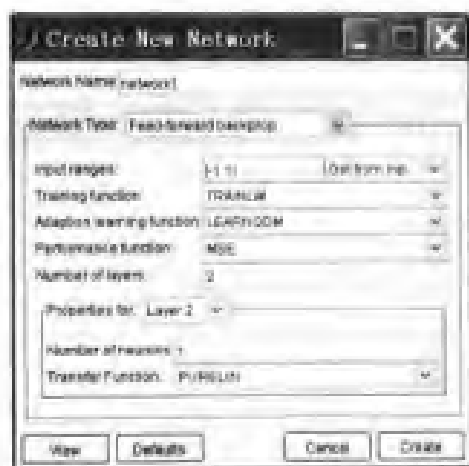


图 9-24 创建新网络的设置对话框

一个按钮,即弹出“Networks: Network”界面,此时可对网络进行各种操作:查看网络结构(View)、对网络进行初始化(Initialize)、模拟(Simulate)、训练(Train)、使网络自适应(Adapt)、查看或修改权值(Weights)。显然,一般先执行初始化、训练,然后才能进行模拟。由于在上一步骤创建网络时,自动对网络进行过初始化,因此,这一步除非想重新初始化以外,一般可以不再执行初始化操作,可直接进行网络训练。

网络训练的界面如图 9-25 所示。网络训练的相应代码为“net = train(net, p, t);”,所以在图 9-25 中应分别设置 Training Data 的 Inputs 和 Targets 为 p 和 t。在 Training Results 的 Outputs 和 Errors 分别默认为 network1_outputs 和 network1_errors,用于保存网络结果输出和网络误差,这里分别改为 training_outputs 和 training_errors。还可以对训练参数进行修改,这里按默认。所有这些设置完成后,按[Train Network]开始训练。训练结束后,弹出一个训练性能图,见图 9-26,由图可见,总残差目标趋近于 0。



图 9-25 网络训练

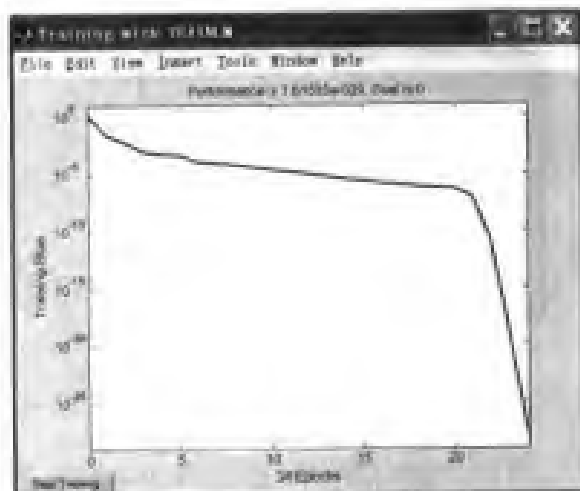


图 9-26 训练性能

训练好后,可点击[Simulate]进行模拟。模拟界面如图 9-27,先设置 Simulation Data 的 Inputs: p(这里以网络输入值 p 作为模拟计算的输入值),再设置 Simulation Results 的 Outputs: yp(输出变量名,默认为 network1-outputs)。设置好后,按[Simulate Network]即开始模拟。模拟结束后,切换到网络/数据管理器,在 Outputs 文本框上即显示“yp”,用鼠标点中它,再

点击[View]按钮，即可查看此模拟结果，如图 9-28 所示。可见，此结果与[例 9-4]完全相同。



图 9-27 网络模拟界面

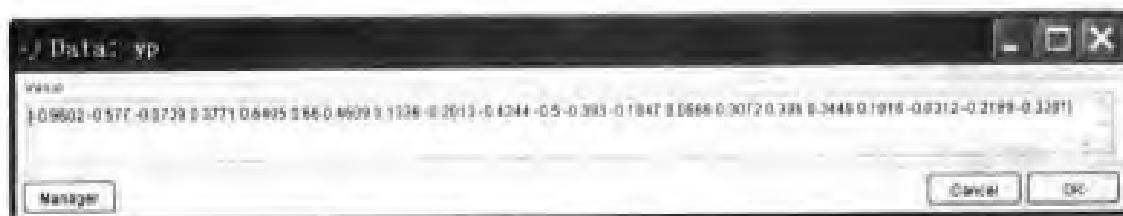


图 9-28 模拟结果

当然，还可模拟计算其他输入值下的网络输出。如计算 $x = [0.55, 0.69]$ 下的输出值 y ，只需返回网络/数据管理器主界面，同前面方法用[New Data...]按钮，将此输入向量 $x = [0.55, 0.69]$ 输入到网络/数据管理器中，然后在如图 9-29 所示的模拟界面设置 Simulation Data 的 Inputs: x ，Simulation Results 的 Outputs: y ，再按[Simulate Network]即可计算对应于输入 $x = [0.55, 0.69]$ 的网络输出 y 的值。

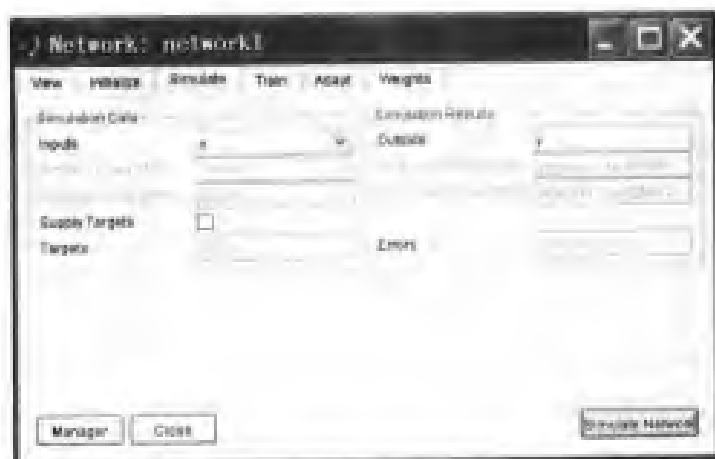


图 9-29 模拟界面

模拟计算结束后，在主界面可观察到 y 的结果如图 9-30 所示。

最后，可在主界面按[Export...], 显示如图 9-31 所示的界面，单击[Select All], 再按[Save]

即可全部保存到指定的磁盘文件中。也可按[Export]将选择的变量输出到 MATLAB 工作环境中，然后执行“>save xNewff_nntool”把所有变量存入 xNewff_nntool.mat 中。这样，下次想再使用此训练好的网络，只需在 MATLAB 工作环境中执行“>load xNewff_nntool”把所有变量装载到存入工作环境中，然后在网络/数据管理器主界面点击[Import...], 即可把已装载的部分或所有变量从工作环境输入到网络/数据管理器中。



图 9-30 对应于 $x=[0.55, 0.69]$ 的网络输出

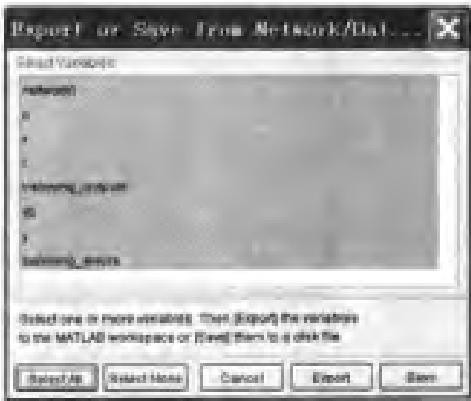


图 9-31 从网络/数据管理器输出或保存变量

9.7 神经网络在化工中的应用实例

【例 9-8】 用神经网络对高效液相色谱（HPLC）柱流动相进行模拟^[5]

酒分析的 HPLC 优化——建立酒样的常规分析规程：HPLC 分析方法的主要指标是分析的效率因子，它涉及各组分在柱中的分离程度以及整个分析所用的时间两方面内容。分离程度和分析时间均受到流动相选择的影响：若它通过柱子太快，则分离效果差；若它通过柱子太慢，则分析时间太长。所以建立分析规程的主要目标是对流动相的效率因子设计一个模型，以寻找最合适的流动相。本例仅考虑流动相两个性质：乙醇浓度 C_e 和流动相的酸度（pH 值）。经验表明， C_e 应在 10%~30% 之间，而 pH 应在 5.0~6.0 之间。见图 9-32。

为便于建模，需通过实验获得实验数据。这里进行两变量三水平的正交实验，测得每一实验条件下的 HPLC 色谱，进而算得相应的效率因子 PF ，实验结果如表 9-1 所示。

表 9-1 用于建立一个 HPLC 过程模型的数据

$C_e / \%$	pH	PF	$C_e / \%$	pH	PF
10	5.0	6.08	30	5.5	3.13
20	5.0	2.42	10	6.0	7.06
30	5.0	2.10	20	6.0	3.72
10	5.5	7.31	30	6.0	3.37
20	5.5	3.00			

模型 采用径向基神经网络。

程序说明 用 newrb() 创建一个径向基神经网络，并用 sim() 模拟。其中实验数据存入 HPLCdata.mat 中供程序调用。在模拟计算时，将 C_e 在 10%~30% 之间及 pH 在 5.0~6.0 之间分别分成 $N = 30$ 个等份，这些离散数据 $[C(j), pH(i)]$ 构成实验条件矩阵。然后用 sim() 模拟计算各个条件 $[C(j), pH(i)]$ 下的效率因子 $PF(i, j)$ ，最后用 max() 找出 PF 最高者。

程序清单 HPLC_RB.m

```
function HPLC_RB
load HPLCdata
p = HPLCdata(:, 1:2)'; t = HPLCdata(:, 3)';
eg = 0.02;          % sum-squared error goal.残差平方和指标
sc = 1;             % spread constant radial basis functions.径向基函数的分布常数
net = newrb(p, t, eg, sc);
% To see how the network performs, replot the training set:
title('Training Vectors'); xlabel('C, %'); ylabel('pH'); zlabel('PF')
% Simulate the network response for inputs over the same range:
PF = sim(net, p); N = 30; C = linspace(10, 30, N); pH = linspace(5.0, 6.0, N);
for i=1:N
    for j=1:N
        PF(i, j) = sim(net, [C(j); pH(i)])
    end
end
PFmax = max(max(PF)); [i, j] = find(PF==PFmax); Copt = C(j); pHopt = pH(i)
PFopt = sim(net, [Copt; pHopt])
% And plot the results on the same graph:
surf(C,pH,PF), xlabel('C, %'); ylabel('pH'); zlabel('PF'), title('模拟结果');
```

计算结果 神经网络模型中的最佳点为 $C_{e_opt} = 10\%$ 和 $pH_{opt} = 5.7$ ，对应的最高效率因子为 $PF_{opt} = 7.41$ 。

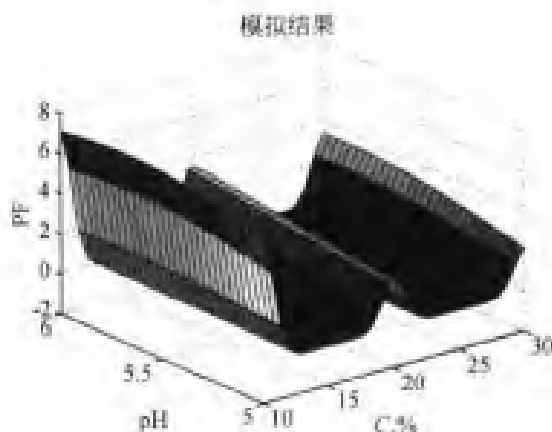


图 9-32

符号说明

标量、向量和矩阵

方程和图中出现的标量、向量和矩阵的数学符号如下。

标量 - 小斜体字母 (a , b , c)

向量 小黑体非斜体字母 (\mathbf{a} 、 \mathbf{b} 、 \mathbf{c})

矩阵 大写字母非斜体字母 (\mathbf{A} 、 \mathbf{B} 、 \mathbf{C})

权矩阵 \mathbf{W} 和阈向量 \mathbf{b}

权标量元素 w_{ij} , i —行, j —列

权列向量 \mathbf{w}_j — 由权矩阵 \mathbf{W} 第 j 列组成的向量

权行向量 \mathbf{w}_i — 由权矩阵 \mathbf{W} 第 i 行组成的向量

阈标量元素 b_i

层符号

上标可来表示变量所在的神经元层。例如,第3层网络输入可表示为 \mathbf{n}^3 。上标还用于识别层权矩阵(layer weight matrices)和输入权矩阵(input weight matrices)之间的源(1)和目标(k)的连接。例如,从第2层到第3层的层权矩阵(layer weight matrix)可表示为 $\mathbf{LW}^{3,2}$ 。

输入权矩阵: $\mathbf{IW}^{k,1}$

层权矩阵: $\mathbf{LW}^{k,1}$

数学符号与 MATLAB 代码符号的对应关系

- 把上标变为单元数组下标,例如: $p^1 \rightarrow p\{1\}$
- 把下标变为圆括号下标,例如: $p_2 \rightarrow p(2)$, $p_2^1 \rightarrow p\{1\}(2)$
- 把圆括号下标变为第2个单元数组下标,例如, $p^1(k-1) \rightarrow p(1, k-1)$
- 把数学操作符转变为 MATLAB 操作符以及 toolbox 函数,例如, $ab \rightarrow a*b$

参 考 文 献

- 1 Howard Demuth and Mark Beale, Neural Network Toolbox User's Guide, Version 4.0. The MathWorks, Inc., 2002
- 2 楼顺天, 施阳编著. 基于 MATLAB 的系统分析与设计——神经网络. 西安: 西安电子科技大学出版社, 2000
- 3 Chen, S., C.F.N. Cowan, and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, March 1991, 302~309
- 4 P.D. Wasserman, *Advanced Methods in Neural Computing*, New York: Van Nostrand Reinhold, 1993, 155~161, 35~55
- 5 J. Zupan, and J. Gasteiger, *Neural Networks for Chemists: Introduction*. VCH, 1993. 中译本: J. Zupan, and J. Gasteiger 著. 潘忠孝, 陈玲然译. 神经网络及其在化学中的应用. 合肥: 中国科学技术大学出版社, 2000

光盘内的程序文件一览表

第2章 MATLAB 编程基础

例 题	程 序 文 件	功 能
例 2-1	<i>xScript.m</i>	脚本文件示例
例 2-2	<i>xFunction.m</i>	用 function 声明函数及函数调用方法
例 2-3	<i>x2Dplot.m</i>	调用数据 M 文件, 并绘制二维曲线图
	<i>Data_x2Dplot.m</i>	数据 M 文件的生成
例 2-4	<i>IntegByEulerpro.m</i>	直接积分
	<i>eulerpro.m</i>	改进欧拉方法的通用积分函数
	<i>xEulerpro.m</i>	调用函数 eulerpro 以求解微分方程
例 2-5	<i>Speedup1.m</i>	避免使用循环, 而使用向量或矩阵代替以提高执行速度
例 2-6	<i>Speedup2.m</i>	在循环语句之前预先对向量、矩阵或数组初始化以提高执行速度
例 2-7	<i>Speedup3.m</i>	分别用自定义函数和 MATLAB 内部函数计算一个矩阵的所有元素之和, 并比较这两种方法的速度
其他	<i>CommandDemo.m</i>	演示 MATLAB 常用命令的基本操作
	<i>xFscanf.m, FileIO.dat</i>	函数 fscanf() 的用法
	<i>xNargin.m</i>	命令 nargin 的用法
	<i>xBreakPause.m</i>	break 和 pause 的用法
	<i>xFplot.m</i>	函数 fplot() 的应用

第3章 MATLAB 在数值分析中的应用

例 题	程 序 文 件	功 能	主 要 函 数
例 3-1	<i>xInterp1.m</i>	一维插值	interp1()
例 3-2	<i>xInterp2.m</i>	二维插值	interp2()
例 3-3	<i>xSpline.m</i>	三次样条函数插值	spline(), interp1()
例 3-4	<i>xPolyfit.m</i>	最小二乘多项式拟合函数	polyfit()
例 3-5	<i>xPolyfitLinear.m</i>	线性最小二乘法拟合	polyfit()
例 3-6	<i>xSplineFit.m</i>	拟合离散数据	csaps(), spap2(), spaps()
例 3-7	<i>FermInterp.m, FermData.m</i>	发酵数据的三次样条插值	spline(), ppval()
例 3-8	<i>FermDataFit.m</i>	发酵数据的最小二乘样条拟合和最小二乘多项式拟合	spap2(), fnval(), polyfit(), polyval()
例 3-9	<i>xIntegration.m</i>	用梯形求积法、自适应 Simpson 求积 (低阶) 法和自适应 Lobatto 求积 (高阶) 法进行数值积分	trapz(), quad(), quadl()
例 3-10	<i>xSingularInt.m</i>	奇异积分	quad(), quadl()
例 3-11	<i>xDifferential.m</i>	用有限差分法、多项式拟合法和三次样条插值方法对离散数据进行数值微分	diff(), polyfit(), polyder(), polyval(), csapi(), fnder(), fnval()
例 3-12	<i>DistStagesCaL.m</i>	计算双组分简单精馏塔的理论板数	quadl(), csaps()
例 3-13	<i>PFRlength.m</i> <i>PFRconv.m</i>	等温管式反应器的管长计算	quadl(), fzero()
例 3-14	<i>PFRSpaceTime.m</i>	等温管式反应器的空时计算	quadl()
例 3-15	<i>AbsPackedHeight.m</i>	计算填料吸收塔总传质单元数 N_{OX}	quadl()

续表

例 题	程 序 文 件	功 能	主 要 函 数
例 3-16	<i>RTD1.m</i>	反应器停留时间分布的混合特性	spaps()
例 3-17	<i>KineticDataFit.m</i>	用微分法进行动力学数据分析	spap2()、fnder()、polyfit()
例 3-18	<i>xFzero_Roots.m</i>	演示如何使用 fzero() 和 root()	fzero()、root()
例 3-19	<i>xFsolve.m</i>	解非线性方程组	fsolve()
例 3-20	<i>PVT.m</i>	由状态方程 P-V-T 关系计算	fzero()
例 3-21	<i>BatchDist.m</i>	简单蒸馏计算—单变量非线性方程	fzero()
例 3-22	<i>BoilingPoint.m</i>	计算混合液体的泡点及平衡蒸汽组成	fzero()
例 3-23	<i>Flash.m</i>	等温闪蒸计算	fzero()
例 3-24	<i>ConvCSTR.m</i>	求解绝热连续搅拌槽反应器的转化率	fzero()
例 3-25	<i>PipeDiameter.m</i>	估算非牛顿型流体流动所需的管道直径	fzero()
例 3-26	<i>ChemEquil.m</i>	化学平衡计算	fsolve()
例 3-27	<i>MultipleStateCSTR.m</i>	求有夹套换热 CSTR 的多稳态解	fsolve()
例 3-28	<i>xODE.m</i>	求 ODE 初值问题	ode45()、ode23()
例 3-29	<i>xODEs.m</i>	求 ODE 系统的初值问题	ode45()、ode23()
例 3-30	<i>xStiffODEs.m</i>	解刚性常微分方程组	ode23s
例 3-31	<i>xBVP1.m</i>	求解两点边值问题	bvp4c()、bvpinit()
例 3-32	<i>xBVP2.m</i>	求解两点边值问题	bvp4c()、bvpinit()

第 4 章 化工中的常微分方程及其求解

例 题	程 序 文 件	功 能	主 要 函 数
例 4-1	<i>BatchReactor.m</i>	间歇反应器中的连串-平行复杂反应系统	ode45()
例 4-2	<i>CSTR.m</i>	CSTR 反应器开车过程的动态模拟和稳态模拟	ode45()
例 4-3	<i>IsothermCSTRs.m</i>	三个串联的 CSTR 等温反应器	ode45()
例 4-4	<i>IsothermTR.m</i>	等温管式反应器	ode45()
例 4-5	<i>NonIsothermTR.m</i>	非等温管式反应器—维稳态拟均相模型的模拟计算	ode45()
例 4-6	<i>SemiBatchReactor.m</i>	半连续反应器—苯的氯化	ode45()
例 4-7	<i>ConDistill.m</i>	连续多组分精馏	ode45()
例 4-8	<i>Absorption.m</i>	气体吸收塔模拟计算	ode45()、spline()
例 4-9	<i>MultiCompDiff.m</i>	多组分气体扩散	ode45()、fminsearch()
例 4-10	<i>DrugAbsorp.m</i>	预测口服药在人体中的吸收	ode45()
例 4-11	<i>PorousPlateCat_1Ord.m</i>	计算片状催化剂中等温一级不可逆反应的有效因子(等温内部效率因子)	bvp4c()
例 4-12	<i>PorousSphereCat_1Ord.m</i>	多孔球形催化剂在等温条件下的一级不可逆反应	bvp4c()、spline()、quadl()
例 4-13	<i>PorousSphereCat_2Ord.m</i>	多孔球形催化剂在等温条件下的二级不可逆反应	bvp4c()、spline()、quadl()
例 4-14	<i>DiffReact_SphereCat.m</i>	多孔球形催化剂在等温条件下的一级不可逆反应	bvp4c()
例 4-15	<i>DiffReact_MonolithCat.m</i>	多孔催化剂层中伴随有等温可逆反应的双组分气体扩散	bvp4c()
例 4-16	<i>EnzymeReactDiff.m</i>	氧扩散进入球形细胞中进行酶催化反应	bvp4c()
例 4-17	<i>DrugDelivery.m</i>	通过对药丸包衣的溶解控制药物释放——固体在溶液中的非稳态溶解	ode45()
例 4-18	<i>ChemHeatPump.m</i>	可逆气固化学热泵制冷系统的动态模拟	ode23s()
例 4-19	<i>NewtonFluidFlow.m</i>	水平管中不可压缩牛顿流体的层流流动	bvp4c()
例 4-20	<i>BatchBR_CellCult.m</i>	厌氧间歇发酵动态模拟	ode45()
例 4-21	<i>BioReactCSTR.m</i>	生化反应器模拟计算	fsolve()、ode45()
例 4-22	<i>MembraneReactor.m</i>	膜催化反应器的模拟计算	ode45()
例 4-23	<i>NonIsothermCSTR.m</i>	非等温 CSTR	ode23s()
例 4-24	<i>CSTRs_Closedloop.m</i>	三个串联 CSTR 等温反应器的闭环控制系统	ode45()
例 4-25	<i>BatchDistill.m</i>	双组分间歇精馏的回流比控制	ode45()

第5章 化工中的偏微分方程及其求解

例 子	程 序 文 件	功 能	主 要 函 数
例 5-2	<i>PDE1Dd_CrankNicolson.m</i> <i>CrankNicolson.m</i> <i>TDMA.m</i>	用 Crank-Nicolson 差分格式求解 1 维动态热传导方程	(全部用自定义函数)
例 5-3	<i>PDEs2DS_CrankNicolson.m</i>	用有限差分法求解固定床反应器 2 维拟均相稳态模型 (2 维稳态 PDE 方程组)	fsolve()
例 5-4	<i>PDEs2DS_Collocation.m</i>	用对称正交配置法求解例 5-7 的问题 (2 维稳态 PDE 方程组)	ode45(), quadl()
例 5-5	<i>PDE1Dd_MOL.m</i>	用 MOL 法求解 1 维动态热传导方程	ode45()
例 5-6	<i>PDE1Dd_pdepe.m</i>	用 MATLAB 的 MOL 法函数 pdepe() 求解 1 维动态热传导方程	pdepe()
例 5-7	<i>pdex4.m</i>	利用 MATLAB 函数 pdepe() 求解 1 维动态 PDE 方程组	pdepe()
例 5-8	<i>PDEsD1ds_Catalyst.m</i>	求解多孔球形催化剂颗粒中的热质传递模型 (1 维动态 PDE)	pdepe()
例 5-9	<i>PDEtoolPoisson.m</i>	用 pdetool 求解 2 维稳态椭圆型 PDE 边值问题 (Poisson 方程)	pdetool
例 5-10	<i>PDEtool2DD_CrackHT.m</i>	用 pdetool 求解 2 维动态热传导方程 (无内热源) — 抛物型 PDE	pdetool
例 5-11	<i>PDEtool2DD_FluidFlow.m</i>	用 pdetool 模拟计算矩形管道中的牛顿型流体流动 (2 维动态 PDE)	pdetool
例 5-12	<i>Poisson.m</i> <i>Poissonb.m, Poissong.m</i>	利用 PDE 工具箱函数求解例 5-9 的泊松方程 (2 维稳态 PDE)	asmpde()
例 5-13	<i>HeatCond_2DD.m</i> <i>HeatCondb.m, HeatCondg.m</i>	求解 2 维动态热传导问题	parabolic()
例 5-14	<i>PDE2DD_FluidFlow.m</i> <i>FluidFlowg.m</i> <i>FluidFlowb.m</i>	利用 PDE 函数通过编程模拟计算矩形管道中的牛顿型流体流动 (问题同例 5-11) 2 维动态 PDE 方程	parabolic(), tri2grid
例 5-15	<i>CylindCat1.m</i> <i>CylindCat1g</i> <i>CylindCat1b</i>	计算等温一级反应的圆柱状催化剂颗粒中的浓度分布和有效因子 (2 维稳态 PDE 方程)	pdenonlin()
例 5-16	<i>CylindCat2.m</i> <i>CylindCat2g.m</i> <i>CylindCat2b.m</i>	计算圆柱状催化剂颗粒中进行一级反应的浓度分布 2 维稳态 PDE 方程	pdenonlin()

第6章 化工最优化方法

例 子	程 序 文 件	功 能	主 要 函 数
例 6-1	<i>xFminbnd.m</i>	单变量函数的最小化	fminbnd()
例 6-2	<i>x1LinProg.m</i>	求解线性规划问题	linprog()
例 6-3	<i>x2LinProg.m</i>	求解线性规划问题	linprog()
例 6-4	<i>xFminsearch.m</i>	用 Nelder-Mead 单纯形法求 Rosenbrock 的“香蕉函数”	fminsearch()
例 6-5	<i>x1QuadProg.m</i>	求解二次规划问题	quadprog()
例 6-6	<i>x2QuadProg.m</i>	求解二次规划问题	quadprog()
例 6-7	<i>x3QuadProg.m</i>	求解二次规划问题	quadprog()
例 6-8	<i>x1Fmincon.m</i>	多变量有约束最优化 (非线性规划) 问题	fmincon()
例 6-9	<i>x2Fmincon.m</i>	多变量有约束最优化 (非线性规划) 问题	fmincon()

续表

例 子	程 序 文 件	功 能	主 要 函 数
例 6-10	<i>xFgoalattain.m</i>	多目标最优化	fgoalattain()
例 6-11	<i>xLsqcurvefit.m</i>	非线性最小二乘曲线拟合	lsqcurvefit()
例 6-12	<i>CoolerOptDes.m</i>	冷却器的最优设计	fminsearch()
例 6-13	<i>HeatExchangerDesign.m</i>	换热器系列的最优设计	fmincon()
例 6-14	<i>CSTR_OptDes.m</i>	连续搅拌槽式反应器的最优设计	fmincon()
例 6-15	<i>TempOpt_BatchReactor.m</i>	计算间歇反应器中连串-平行复杂反应系统的最优反应温度	fminsearch()
例 6-16	<i>PFRTempOpt_A.m</i>	计算管式反应器的最优温度分布	ode45()
	<i>PFRTempOpt_B.m</i>		ode45()、fmincon()
	<i>PFRTempOpt_C.m</i>		bvp4c()、fmincon()
例 6-18	<i>LinearProg.m</i>	生产规划(线性规划问题)	linprog()

第 7 章 参数估计和模型辨识

例 子	程 序 文 件	功 能	主 要 函 数
例 7-1	<i>KineticsEst1_Diff.m</i>	用微分法和积分法确定反应速率方程	lsqnonlin()
	<i>KineticsEst1_Int.m</i>		lsqnonlin()、ode45()
例 7-2	<i>KineticsEst2.m</i>	利用非线性最小二乘法估计动力学参数	lsqnonlin()
例 7-3	<i>KineticsEst3.m</i>	利用非线性最小二乘法估计动力学参数	regress()、lsqnonlin()
例 7-4	<i>KineticsEst4.m</i> <i>ChemKineticsData.mat</i>	利用非线性最小二乘法估计动力学参数	regress()、lsqnonlin()
例 7-5	<i>KineticsEst5.m</i> <i>KineticsData1.m</i> , <i>Output.m</i>	等温间歇反应器中复杂反应系统的动力学参数估计(ODE 模型)	fmincon()、lsqnonlin()
例 7-6	<i>KineticsEst6.m</i> <i>KineticsData2.m</i>	利用非线性最小二乘法估计反应系统 ODEs 模型的动力学参数	lsqnonlin()
例 7-7	<i>EnzymeKinetEst.m</i>	酶催化反应动力学(代数模型)的参数估计	regress()、lsqnonlin()
例 7-8	<i>BioKineticsEst.m</i>	生化反应动力学参数估计	regress()、lsqnonlin()
例 7-9	<i>FermKineticsEst.m</i> <i>Weights.m</i> <i>FermKineticsData.mat</i>	利用非线性加权最小二乘法估计青霉素发酵过程动力学参数(ODEs 模型)	lsqnonlin()、ode45()
例 7-11	<i>VLEfitting.m</i>	用非线性最小二乘法估计二元混合物的 van Laar 模型中的参数	Lsqnonlin()
例 7-12	<i>KineticsEst_Ident1.m</i> <i>rho2_F.m</i>	判断例 7-4 的模型是否适应	regress()、lsqnonlin()
例 7-13	<i>KlneticsEst_Ident2.m</i>	模型辨识	Lsqnonlin()

第 8 章 化工实验设计及数据处理

例 子	程 序 文 件	功 能	主 要 函 数
例 8-1	<i>OrthExpAnalysis1.m</i>	无交互作用的正交实验设计及其实验结果的极差分析	sum()、find()、max()、min()
例 8-2	<i>OrthExpAnalysis2.m</i>	有交互作用的正交实验设计及其实验结果的极差分析	sum()、find()、max()、min()
例 8-3	<i>OrthExpAnalysis3.m</i>	对例 8-1 进行方差分析	sum()、find()、size()、length()
例 8-4	<i>OrthExpAnalysis4.m</i>	有交互作用的正交实验设计及其实验结果的方差分析	sum()、find()、size()、length()
例 8-8	<i>DOE.m</i>	用于动力学参数估计的 D-最优试验设计	cordexch()、nlinfit()

第9章 神经网络在化工中的应用

例 子	程 序 文 件	功 能	主 要 函 数
例 9-1	<i>xNewlind.m</i>	设计一个单层线性神经网络	newlind(), sim()
例 9-2	<i>xLinPred.m</i>	利用线性网络进行线性预测	newlind(), sim()
例 9-3	<i>xAdapLinPred.m</i>	利用线性网络进行自适应预测	newlin(), adapt()
例 9-4	<i>xNewff.m</i>	应用两层 BP 网络来完成函数逼近任务	newff(), sim()
例 9-5	<i>xNewrb.m</i>	应用径向基网络来完成函数逼近任务	newrb(), sim()
例 9-6	<i>xNewgrnn.m</i>	用普遍化回归神经网络 (GRNN) 实现函数逼近	newgrnn(), sim()
例 9-7	<i>xNewff_nntool.mat</i>	用 nntool 求解例 9-4 的相同问题	nntool
例 9-8	<i>HPLC_RB.m</i> <i>HPLCdata.mat</i>	用神经网络对高效液相色谱 (HPLC) 柱流动相进行模拟	newrb(), sim()

